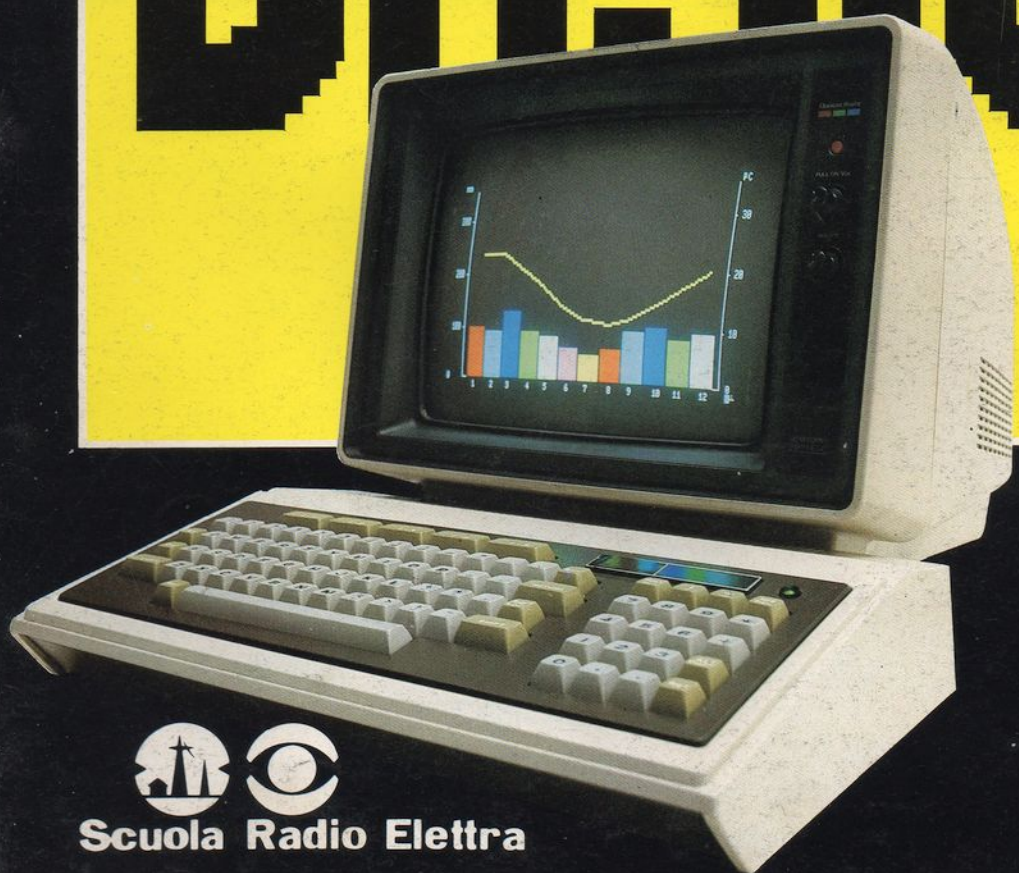
PARLA BASIC



Scuola Radio Elettra

Premessa

Nei precedenti tre gruppi di lezioni sono state fornite le nozioni preliminari e fondamentali dell'informatica e della microinformatica, con particolare riferimento alle applicazioni sui personal computer.

Inoltre, sono stati trattati i primi elementi della programmazione BASIC, *che è il linguaggio piú adatto e generalmente piú diffuso per coloro che desiderano introdursi nel settore dell'elaborazione elettronica digitale.*

Nei tre gruppi di lezioni che seguiranno, oltre ad acquisire ulteriori nozioni riguardanti il linguaggio BASIC, *Lei approfondirà il concetto di programmazione ed apprenderà ad elaborare in modo piú autonomo diagrammi di flusso e linguaggi di programmazione attraverso esercizi e problemi operativi.*

Scuola Radio Elettra
Torino

Analisi e programmazione

Lezione 4

1 — Il linguaggio di programmazione BASIC
2 — Gli enunciati BASIC
3 — Le prime istruzioni BASIC
4 — I primi comandi BASIC
5 — Il controllo del programma
6 — Il controllo del cursore e dello schermo

Che cosa apprenderà

In questa lezione amplierà le Sue conoscenze sul ruolo svolto dalle prime istruzioni BASIC (**PRINT, INPUT, LET,** ecc.) e dai primi comandi BASIC (**NEW, LIST, SAVE, VERIFY, RUN,** ecc.).

Inoltre, avrà modo di apprendere come si deve operare quando un programma è in fase di elaborazione, per interromperlo in qualsiasi momento, correggerlo ed inserirvi altre istruzioni.

1. – Il linguaggio di programmazione BASIC

In precedenza si è precisato che un programma BASIC è costituito sostanzialmente da un insieme di istruzioni numerate.

Ogni istruzione BASIC, a sua volta, è composta da tre parti: da un **numero di riferimento**, da una **parola chiave** o **riservata** (una parola dell'alfabeto inglese) e da una **espressione algebrica** o **alfanumerica**, che si definisce anche "oggetto di elaborazione".

Come già precedentemente accennato, i numeri di riferimento sono in ordine naturale crescente, ma non necessariamente crescenti di uno in uno.

Le parole chiavi sono parole inglesi riservate soltanto per scopi mnemonici e servono a definire il tipo di operazione che deve essere eseguita.

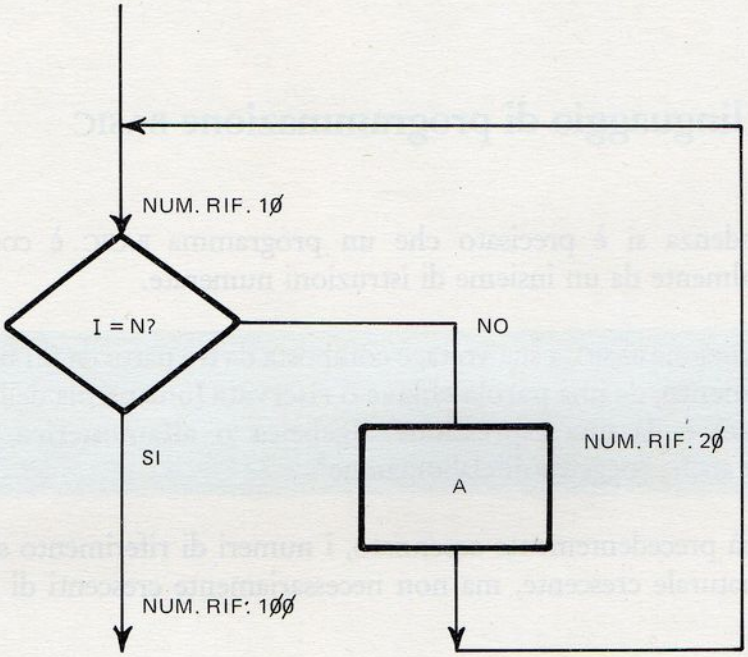
L'espressione "oggetto di elaborazione", che è posta a destra subito dopo la parola chiave, può essere o un'espressione algebrica vera e propria oppure un'espressione alfanumerica racchiusa tra virgolette, oppure ancora può contenere altre parole chiavi o numeri di riferimento di altre istruzioni, ecc.

Dipendentemente dal sistema di elaborazione, la struttura del linguaggio BASIC può essere piú o meno complessa; ciò si riflette sulle prestazioni piú o meno semplici che il linguaggio può offrire.

Cenni di programmazione strutturata

Per esempio, nei sistemi personal computer con capacità di memoria superiori a 48 Kbyte, il linguaggio BASIC consente, a differenza di altri microsistemi, l'effettuazione della programmazione strutturata, che è un insieme di tecniche compatte, cioè, alcune istruzioni del linguaggio BASIC condensano e racchiudono in passi di elaborazione piú compatti quello che prima era frammentato in piú istruzioni.

Una tipica istruzione strutturata è la seguente: **WHILE... WEND**. Questa istruzione sostituisce, quando il sistema lo consente, il complesso delle istruzioni **IF... THEN...** che, come si è visto nella precedente lezione, realizzano la seguente alternativa logica:



La struttura di questa alternativa semplice, se realizzata con l'istruzione **IF... THEN...**, verrebbe programmata nel seguente modo:

```
10 IF I=N THEN 100
20 .....
30 .....
⋮
70 GOTO 10
100.....
110.....
120.....
⋮
200.....
⋮
```

blocco A

Se realizzata invece con l'istruzione strutturata **WHILE... WEND**, tale

alternativa verrebbe programmata nel seguente modo piú compatto (e piú comprensibile):

```
10 WHILE I=N
20 .....
30 .....
:
70 WEND
100 .....
110 .....
```

blocco A

In pratica si eliminano le due istruzioni di salto, una contenuta nell'istruzione 10 cioè:

```
10 IF I=N THEN 100 ← salto a 100
```

e l'altra contenuta nell'istruzione 70:

```
70 GOTO 10
```

La programmazione strutturata consente di organizzare il programma in maniera piú organica e comprensibile, perché fa scarso uso di salti e deviazioni che, a volte, possono far perdere al programmatore il filo logico della sequenza di elaborazione.

Ad ogni numero di riferimento corrisponde "almeno" un'istruzione BASIC (individuata dalla parola chiave), cioè possono esserci piú istruzioni che corrispondono ad uno stesso numero. Per esempio, la sequenza di interlinee che consente di spaziare sullo schermo o sulla stampante due linee di stampa può essere cosí programmata:

```
:
5 PRINT "SCUOLA RADIO ELETTRA"
10 PRINT: PRINT: PRINT: PRINT
15 PRINT "TORINO"
:
```

Al numero 10 della lista di programma riportata sopra vengono eseguite quattro interlinee o spaziature tra la prima istruzione di stampa (5) e la seconda istruzione di stampa (15) e queste interlinee vengono effettuate in

maniera compatta, scrivendo quattro istruzioni **PRINT** al numero 10, separate ognuna dal carattere : (due punti).

Naturalmente non soltanto l'istruzione **PRINT** può essere ripetuta su una stessa linea.

Per esempio, si può riscrivere la stessa lista nel seguente modo, piú chiaro e per cosí dire "commentato":

Istruzione **PRINT** con "commento"

```
5 PRINT "SCUOLA RADIO ELETTRA": REM STAMPA
PRIMA ESPRESSIONE ALFABETICA
10 PRINT:PRINT:PRINT:PRINT: REM STAMPA
QUATTRO INTERLINEE
15 PRINT "TORINO": REM STAMPA LA SECONDA
ESPRESSIONE ALFABETICA
```

Come si vede, ogni istruzione della lista precedente è stata commentata da un'istruzione di commento appropriata **REM**.

Istruzione di commento **REM**

Questa istruzione, come è noto, ha la funzione di descrivere il programma e non ha alcun valore operativo; essa serve soltanto a ricordare, a chi programma, quale tipo di elaborazione sta eseguendo in quel momento il computer.

Quanto può essere lunga un'istruzione con un certo numero di riferimento?

Se si eprime la lunghezza di un'istruzione con il numero di caratteri, questa lunghezza varia da computer a computer: nei microsistemi essa non può superare 80 caratteri, mentre nei personal computer di maggiore capacità arriva di solito a 256 caratteri, comprensivi della parola chiave e del numero di riferimento.

Per esempio, nell'istruzione 5 della lista di programma precedente, dopo la parola chiave **PRINT**, si sarebbe potuto scrivere un'espressione letterale, racchiusa tra virgolette, lunga al massimo 256 caratteri alfanumerici, compresi gli spazi e le virgolette.

Si è anche visto che i numeri di riferimento possono essere inseriti in modo crescente, ma distanziati di 5 in 5 o di 10 in 10.

Come già precedentemente accennato, viene adottato questo sistema per dare la possibilità al programmatore di inserire altre istruzioni, dove il

programma lo richieda, tra un numero ed il successivo.

Nella lista di programma precedente, tra l'istruzione 1Ø e l'istruzione 15, per esempio, si potrebbe introdurre la stampa del nominativo della via dove risiede la SCUOLA RADIO ELETTRA, battendo semplicemente, anche dopo l'istruzione 15:

```
13 PRINT "VIA STELLONE 5"
14 PRINT: PRINT
```

Eseguendo il comando **LIST**, il programma precedente riapparirà sullo schermo in questa veste piú completa:

```
5 PRINT "SCUOLA RADIO ELETTRA": REM STAMPA
PRIMA ESPRESSIONE ALFABETICA
1Ø PRINT:PRINT:PRINT:PRINT: REM STAMPA
QUATTRO INTERLINEE
13 PRINT "VIA STELLONE 5"
14 PRINT: PRINT
15 PRINT "TORINO": REM STAMPA LA SECONDA
ESPRESSIONE ALFABETICA
```

Come si può rilevare, il comando **LIST** inserisce opportuni spazi tra i numeri di riferimento e le parole chiavi.

Eseguendo il programma ora visto con il comando **RUN**, le scritte appariranno sullo schermo nel modo sotto indicato:

```
SCUOLA RADIO ELETTRA
        ↑
4 interlinee
        ↓
VIA STELLONE 5
        ↓
2 interlinee
        ↑
TORINO
```

Valore massimo di una linea di programma

Quali e quanti possono essere i numeri di riferimento in un programma? Anche in questo caso la risposta dipende dal sistema personal computer che si adotta: se quest'ultimo è tale da accettare un linguaggio BASIC abbastanza evoluto, possono essere numerate istruzioni fino al numero 65.535.

Naturalmente non è detto che possa essere introdotto un programma costituito da 65.535 istruzioni, anzi il numero delle istruzioni è di gran lunga inferiore.

Si tenga presente che il programma BASIC, una volta battuto, risiede nella memoria principale e occupa una zona piú o meno vasta di byte, a seconda del numero di istruzioni che si introducono.

Quanto piú lungo è il programma, tanto minore è lo spazio di memoria che rimane a disposizione per l'elaborazione e la memorizzazione delle variabili.

Si è detto che il programma scritto in linguaggio BASIC viene "interpretato" e successivamente eseguito.

Gli errori di sintassi

La fase di interpretazione e susseguente esecuzione viene compiuta dopo che il programma è stato introdotto nella memoria principale e corretto da eventuali errori di sintassi (scorrettezze di tipo formale, che vengono segnalate dal sistema attraverso adeguati messaggi).

Per esempio, l'istruzione errata:

```
10 PRIMT
```

è segnalata, quando il programma viene eseguito, con il messaggio:

```
? SYNTAX
  ERROR IN 10
```

Oltre agli errori di sintassi possono esistere altri tipi d'errore, i quali però vengono segnalati durante l'esecuzione del programma; tra questi ci sono gli errori logici che sono dovuti al malfunzionamento dell'elaborazione.

Questi errori sono difficili da individuare e correggere: generalmente, conviene rivedere sempre il programma sin dalle prime fasi della sua scrittura, iniziando cioè dall'analisi (algoritmo).

Se per esempio il computer è stato programmato per eseguire il calcolo della lunghezza della circonferenza con la seguente istruzione (errata, logicamente):

```
10 LC=3.14159*R
```

il risultato che viene visualizzato non può essere la lunghezza di una circonferenza di raggio R, perché ci si è dimenticati che l'esatta formula per determinare tale lunghezza è:

LC=2*3.14159*R $(2\pi r)$

Quindi **non è detto che il computer, pur non segnalando alcun errore, fornisca sempre risultati validi.**

Se in corrispondenza ad un numero di riferimento si inseriscono piú istruzioni (nel qual caso ogni istruzione va separata dalla precedente con il carattere :), è abbastanza laborioso successivamente correggere le istruzioni stesse; se invece si programma un'istruzione per ogni numero di riferimento, è molto agevole correggerla o eliminarla completamente.

Per eseguire quest'ultima operazione, è sufficiente battere il numero di riferimento e subito dopo premere il tasto **RETURN**; per esempio, per eliminare l'istruzione 14 **PRINT:PRINT** della lista di programma precedentemente considerata, è sufficiente battere sulla tastiera:

14 **RETURN**.

Istruzioni per "cancellare" altre istruzioni

In alcuni sistemi personal computer viene utilizzato il comando BASIC **DEL** (abbreviazione di DELETE, che significa "cancella") per eliminare una o piú istruzioni.

Il comando **DEL** oppure **DELETE** funziona come il comando **LIST**; per cancellare ad esempio l'istruzione 5Ø1 basta eseguire il comando:

DEL 5Ø1

Per eliminare la parte di programma che va dall'istruzione 3Ø all'istruzione 85, basta eseguire il comando:

DEL 3Ø, 85

Per correggere qualche carattere o gruppo di caratteri in una istruzione, ogni sistema personal computer utilizza un suo criterio particolare.

L'editing di un programma

Tutto il complesso di regole, comandi o istruzioni, operazioni da tastiera che consente di introdurre il programma, di listarlo, di correggerlo, di registrarlo, ecc. si chiama EDITING (dall'inglese *to edit*, che significa scrivere, stampare, compilare).

L'editing dei personal computer di una certa capacità di memoria può essere costituito da un programma piú o meno complesso, che facilita notevolmente al programmatore il lavoro di compilazione; tale programma si chiama anche EDITOR.

Occorre tenere sempre presente che in un programma non vi possono essere due istruzioni con lo stesso numero di riferimento, in quanto il computer, fra le due, ricorda soltanto l'ultima che è stata immessa.

Inoltre, poiché le istruzioni di un programma rappresentano la sequenza logica degli ordini che vengono impartiti al computer, affinché esso possa svolgere un determinato compito la successione della numerazione delle istruzioni stesse deve avvenire con la medesima progressione logica.

2. – Gli enunciati BASIC

Significato di enunciato BASIC

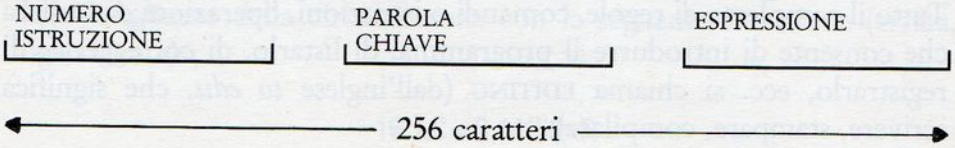
Qualsiasi istruzione o comando BASIC si può considerare un "enunciato", cioè un'espressione alfanumerica (composta di numeri, lettere e caratteri) piú o meno lunga che ha l'effetto quasi sempre di eseguire delle elaborazioni (queste ultime specificate ed indicate nell'enunciato stesso).

In ogni enunciato, si è detto, vengono eseguite quasi sempre delle elaborazioni; ci sono però istruzioni con tutt'altre funzioni: realizzano per esempio delle pause, eseguono la visualizzazione dei colori, generano suoni e note musicali, provvedono all'immissione, all'emissione e all'archiviazione di gruppi di dati, ecc.

Ogni enunciato ha un formato standard: ha un numero di riferimento che serve ad etichettare l'istruzione (numero che non supera generalmente, come si è visto, il valore 65.535), ha una parola chiave (inglese) che serve come codice mnemonico del tipo di operazione, ha un'espressione piú o meno lunga che costituisce l'oggetto dell'elaborazione dell'istruzione.

Formato e struttura di un'istruzione BASIC

CONFIGURAZIONE DI UNA ISTRUZIONE BASIC:



Il numero di riferimento o d'istruzione (che si chiama anche numero della linea di programma), come già visto, può anche mancare: in tal caso l'istruzione è chiamata ad esecuzione immediata e viene eseguita non appena si preme il tasto **RETURN** .

Usando istruzioni ad esecuzione immediata (come precisato nelle precedenti lezioni, non tutte le istruzioni BASIC sono tali), il computer si comporta come una comune calcolatrice tascabile: ogni calcolo algebrico viene eseguito "immediatamente" dopo essere stato introdotto o impostato.

Ad esempio, per calcolare e visualizzare il risultato dell'area di un cerchio con raggio R=2.5 è sufficiente eseguire l'istruzione immediata:

PRINT 3.14159*2.5↑2

e premere ovviamente il tasto **RETURN** .

Sul video o monitor si vedrà apparire il risultato: 19.6349375. Anche in questo caso, se battendo l'istruzione in questione si commette un errore di tipo formale, il personal computer segnala l'errore con il messaggio:

```
? SYNTAX
  ERROR
```

Esempi di errori di sintassi

Nell'elenco che segue sono riportate a sinistra alcune istruzioni **PRINT** ad esecuzione immediata contenenti errori formali, in presenza dei quali il computer si "rifiuta" di operare, oppure fornisce risultati non previsti; le stesse istruzioni sono scritte a destra in forma corretta.

FORMA ERRATA	FORMA CORRETTA
1) **PRIMD** 3.4 * 5	1) **PRINT** 3.4 * 5
2) **P RINT** "A"	2) **PRINT** "A"
3) **PRINT** SCUOLA	3) **PRINT** "SCUOLA"
4) **PRINT** AREA = 3.14159 * R↑2	4) **PRINT** "AREA ="; 3.14159 * R↑2
5) **PRINT** 3,14159	5) **PRINT** 3.14159
6) **PRINT** 4 * 8, 3 * 9, 6 * 9,8	6) **PRINT** 4 * 8, 3 * 9, 6 * 9.8
7) **PRINT** 8 * (3 * 9↑2	7) **PRINT** 8 * (3 * 9↑2)
8) **PRINT** 3,14159	8) **PRINT** "3,14159"

Nella colonna scritta in forma errata sono stati commessi i seguenti errori.

Nell'istruzione 1) è stata battuta la lettera M al posto della N e la lettera D al posto della T della parola PRINT; nell'istruzione 2) è stata frapposta una spaziatura tra la P e la R della parola chiave PRINT; nell'istruzione 3) l'espressione alfabetica non è stata racchiusa tra virgolette (in genere ogni espressione, anche se si tratta di un numero che non ha alcun valore operativo, va racchiusa tra virgolette, allo scopo di consentire una semplice visualizzazione sullo schermo); nell'istruzione 4) si è mescolata un'espressione alfabetica (AREA =) con un'espressione algebrica a carattere operativo; nella forma corretta è stata riscritta esattamente l'istruzione, separando le due espressioni con il carattere ; (punto e virgola).

L'istruzione 5) è errata perché un numero decimale si rappresenta con il punto (.) anziché con la virgola; d'altra parte, questa istruzione viene egualmente accettata dal BASIC, perché viene interpretata come la visualizzazione dei due numeri 3 e 14159 sullo schermo in forma tabulata: ad esempio, nel computer VIC-20 il numero 3 compare a partire dalla seconda colonna o margine sinistro dello schermo ed il numero 14159 a partire dalla tredicesima colonna.

Per tabulare dei numeri sullo schermo o sulla stampante occorre inserire appunto delle virgole tra un dato e l'altro e la spaziatura di stampa è un numero fisso di caratteri, che dipende dalle dimensioni dello schermo o della riga di stampa oppure dal particolare livello del linguaggio BASIC.

Nell'istruzione 6), siccome l'intenzione è di calcolare tre prodotti (di cui l'ultimo: 6 * 9,8) e di tabulare i tre risultati, bisognava scrivere il numero 9,8 con il punto (9.8) anziché con la virgola. Nella istruzione 7) è stata omessa la parentesi chiusa dopo l'esponente 2; infine nell'istruzione 8), che si prefigge di stampare semplicemente il numero 3,14159, non potendosi scrivere la virgola, si doveva racchiudere tra virgolette tutto il numero 3,14159, compresa la virgola.

Finora si è parlato degli enunciati BASIC ad esecuzione immediata.

Le stesse precisazioni fatte in proposito valgono anche per le istruzioni ad esecuzione differita, cioè facenti parte di un programma; in questo caso gli errori formali di sintassi vengono segnalati in fase di esecuzione del programma stesso e la segnalazione riporta il numero dell'istruzione in cui è stato commesso l'errore.

Naturalmente gli errori logici (detti anche errori di programmazione) o non sono segnalati – ed allora deve essere il programmatore a ricontrollare dall'inizio tutte le varie fasi della programmazione – o sono segnalati al momento dell'esecuzione; anche in questo caso è difficile risalire alla causa che ha generato certi risultati errati. Il motto "l'esperienza insegna" è quindi quanto mai valido nel caso della programmazione.

Oltre alle istruzioni BASIC, esistono tipi di enunciati denominati COMANDI. Mentre le istruzioni eseguono operazioni logico-aritmetiche su numeri e parole, i comandi istruiscono il sistema nella gestione delle sue risorse, le quali, come si è visto, sono rappresentate: dall'unità centrale di elaborazione (CPU), costituita dal microprocessore; dalla memoria principale (o memoria RAM ad accesso diretto); dalle unità periferiche, consistenti quasi sempre in una tastiera o keyboard (che serve per l'immissione delle informazioni) e in un monitor o apparecchio TV (che serve per la visualizzazione dei risultati d'elaborazione).

Oltre a queste ultime, i sistemi personal computer, soprattutto i tipi piú recenti, sono dotati o, come si suol dire, accessoriati di altre periferiche, tra cui assumono grande importanza le unità di memoria ausiliaria come le unità a disco e le unità a cassette-nastro magnetico, con le quali è possibile effettuare elaborazioni di una certa complessità.

Le unità di memoria ausiliaria (memorie di massa) sono in grado di archiviare quantità piuttosto ingenti di informazioni, come si è visto nella precedente lezione.

Per gestire un certo numero di unità periferiche è quindi necessario che il sistema sia dotato di un software adeguato al trattamento delle informazioni, ossia di un insieme di comandi od istruzioni operative. Tali comandi si scrivono come le normali istruzioni del linguaggio BASIC ed hanno normalmente effetto diretto, nel senso che sono **ad esecuzione immediata**, cioè non fanno parte di un programma.

Alcuni di questi comandi, quali:

LIST, SAVE, LOAD, VERIFY, RUN, ecc.

sono già stati esaminati nelle lezioni precedenti.

Certi sistemi personal computer, dotati di linguaggio BASIC piú evoluto, consentono la scrittura di tali istruzioni anche all'interno di un programma; ciò risulta utile quando dall'interno di un programma occorre

effettuare registrazioni o letture di dati su un'unità di memoria di massa, oppure smistare la visualizzazione di alcuni risultati su una stampante collegata, oppure passare il controllo dell'elaborazione ad un altro programma che sia stato preventivamente immesso nella memoria principale da una unità collegata, ecc.

Tra i comandi BASIC si distinguono comandi del sistema operativo, quali:

LIST, SAVE, LOAD, VERIFY, CATALOG, NEW, RUN, ecc.

e comandi o istruzioni operative di programma, come le seguenti:

OPEN, INPUT, PRINT, CLOSE, READ, WRITE, ecc.

I comandi del sistema operativo gestiscono globalmente tutte le unità del sistema, mentre le istruzioni operative servono ad effettuare soprattutto, dall'interno di un programma, operazioni particolari di lettura/scrittura su unità di input/output.

3. – Le prime istruzioni BASIC

Nelle precedenti lezioni sono state trattate le istruzioni piú elementari, quali: **PRINT, LET, INPUT,** ecc.

Queste istruzioni, presenti in tutti i tipi di linguaggio BASIC dai meno evoluti ai piú evoluti, sono sufficienti per compiere piccole elaborazioni aritmetiche che non richiedono l'impiego di grandi quantità di dati o che non eseguono operazioni complesse. In questo capitolo, oltre a fornire qualche ulteriore precisazione su queste istruzioni, verranno enunciate alcune istruzioni che riguardano il controllo funzionale dell'elaborazione di un programma.

Innanzitutto bisogna notare che, quando si inizia ad inserire nella memoria principale del computer le istruzioni di un programma, queste vanno a depositarsi in una zona ben delimitata, chiamata **memoria libera dell'utente**, cioè in una zona di memoria riservata al programma, ai dati di input ed ai risultati elaborati; questa zona di memoria è compatta e va da un certo indirizzo minimo ad un indirizzo massimo di memoria.

Alcuni sistemi consentono con appropriate istruzioni di stabilire libera-

mente il minimo e il massimo della memoria principale da utilizzare per il programma; in genere però è bene che sia il sistema operativo del personal computer a delimitare la zona di memoria a disposizione dell'utente, la quale dipende dalla capacità totale della memoria principale.

Comunque il programmatore, specialmente se alle prime armi, non deve preoccuparsi di ciò, ossia dove vanno a finire le istruzioni ed i dati che batte di volta in volta sulla tastiera del computer; deve solamente accertarsi che, prima di iniziare ad inserire le istruzioni, la memoria a lui riservata sia sgombra da altre informazioni (che potrebbero essere costituite da dati e programmi precedentemente inseriti).

A questo scopo viene utilizzato lo speciale comando BASIC **NEW**, che, quando viene eseguito, cancella tutto quello che era presente nella memoria utile.

Dopo questo comando, il programmatore può inserire le varie istruzioni di un programma, perché la memoria a sua disposizione è stata ripulita con il comando **NEW**.

Una volta inserito il programma, istruzione dopo istruzione, egli può eseguirlo oppure, per precauzione, registrarlo nell'unità a cassette, in modo da poterlo riutilizzare successivamente.

In merito alle prime istruzioni BASIC esaminate nelle precedenti lezioni, quali la **PRINT** e la **INPUT**, è bene aggiungere che, a seconda dei sistemi di elaborazione personal computer adottati, queste semplici istruzioni possono assumere varianti formali che le rendono piú efficaci per alcune applicazioni.

Per quanto riguarda l'istruzione **PRINT**, quando non è seguita da altri simboli, essa serve per visualizzare sullo schermo TV i valori di una certa variabile o costante, oppure un'espressione alfanumerica.

Se invece dopo la parola chiave PRINT si aggiunge un simbolo particolare come # (denominato cancelletto oppure diesis) seguito da un numero, la visualizzazione o stampa viene trasferita ad un'unità periferica stampante.

Lo stesso dicasi per l'istruzione di immissione **INPUT**.

Ad esempio, se si vogliono introdurre dei dati non piú dalla tastiera dell'unità centrale ma dalla tastiera di un terminale locale, occorre

scrivere l'istruzione **INPUT** in quest'altro modo:

```
30 INPUT #3, A
```

L'istruzione 3Ø ora vista consente l'immissione di un dato numerico (chiamato A) dall'unità periferica n. 3.

Analogamente l'istruzione di programma:

```
40 PRINT # 1, "SCUOLA RADIO ELETTRA"
```

consente la stampa dell'espressione stringa:

```
"SCUOLA RADIO ELETTRA"
```

sull'unità stampante n. 1.

I numeri che nelle istruzioni precedenti sono riportati dopo il carattere diesis (#) rappresentano le unità periferiche e sono assegnati al momento dell'assemblaggio circuitale del sistema personal computer.

La stampa "personalizzata"

Alcuni sistemi consentono stampe o visualizzazioni piú varie o personalizzate. A tal fine vengono usate istruzioni **PRINT** con l'opzione **USING...**, lista.

Se per esempio si scrive:

```
10 PRINT USING 30, 3.141592, -318.34, -89.10
```

```
3 spazi        3 spazi        3 spazi
30 ↓ ##·## ↓ ####·# ↓ ###·###
```

l'istruzione 3Ø è chiamata anche "maschera" di stampa e configura la modalità di rappresentazione della lista di numeri o espressioni riportata dopo la parola riservata USING, con spaziature diverse da quelle standard. In questo esempio l'immagine dei tre numeri decimali è costituita da tanti caratteri diesis (#) quante sono le cifre che costituiscono la parte intera del numero piú uno, dal punto decimale e da un insieme di altri simboli diesis pari al numero di cifre decimali che si desidera visualizzare sullo schermo.

I valori decimali 3.141592, –318.34, –89.1Ø appariranno quindi sullo schermo nel seguente ordine:

```
3 spazi   3 spazi   3 spazi
↓ 3.14 ↓ -318.3 ↓ -89.100
```

Come si può rilevare, il numero 3.141592 è stato decurtato di quattro cifre decimali, perché la maschera di stampa prevede in questo caso soltanto due cifre decimali.

In genere, perciò, le istruzioni di input/output come la **PRINT**, la **INPUT**, ecc., a seconda del sistema personal computer, possono differenziarsi dalla forma standard.

Inoltre il carattere , (virgola), che ricorre nell'espressione oggetto dell'istruzione **PRINT**, serve a **spaziare un valore da un altro di un certo numero di spazi dipendente anch'esso dal sistema personal computer (cioè dal numero di colonne dello schermo TV).**

La stampa di valori numerici distanziati

Per esempio, se si esegue l'istruzione immediata:

```
PRINT 3 * 6, 4 * 8, 2 ↑ 2
```

dovranno apparire sulla riga dello schermo i valori:

```
n° fisso di caratteri   n° fisso di caratteri   n° fisso di caratteri
┌──────────────────┐    ┌──────────────────┐    ┌──────────────────┐
18                      32                      4
```

distanziati di un numero "fisso" di spazi, numero che dipende dalla larghezza in caratteri della finestra dello schermo TV. Nel caso ora preso in esame sono riservati tre intervalli di uguale lunghezza per i tre numeri.

Ad esempio, nel microcomputer VIC-20, avendo a disposizione soltanto 22 colonne dello schermo TV, il numero 4 dell'istruzione prima considerata verrà visualizzato al di sotto del numero 18.

Come precisato in precedenza, la virgola interposta tra un numero ed un altro consente di spaziare questi ultimi di un certo numero di spazi che dipende dal sistema personal computer.

La stampa di valori adiacenti (non distanziati)

Il carattere punto e virgola (;) permette invece di stampare **due numeri od espressioni qualsiasi immediatamente uno di seguito all'altro, cioè senza spaziatura.**

Per esempio, l'istruzione:

```
PRINT "3.14" ; "159"
```

visualizzerà sullo schermo il seguente numero:

3.14159

così come l'istruzione:

spazio
PRINT "INTRODUCI↓";"N"

farà apparire sullo schermo l'espressione:

INTRODUCI N

Se invece il carattere ; (punto e virgola) viene utilizzato per la stampa di dati numerici non racchiusi fra virgolette, il calcolatore inserirà automaticamente due spazi fra un numero e l'altro.

Ad esempio l'istruzione:

PRINT 3.14 ; 159

farà comparire sullo schermo:

3.14 159

Per quanto riguarda l'istruzione **INPUT**, oltre a quanto detto nella precedente lezione, occorre aggiungere che questa istruzione, la quale serve per l'immissione dei dati da tastiera, può essere scritta in forma "specificata", cioè dopo la parola chiave "INPUT" e prima di scrivere la lista dei nomi delle variabili viene inserita una stringa che ha la funzione di descrivere il tipo di dato da inserire.

Questa stringa apparirà sul video prima del punto interrogativo (?), che è il simbolo usato per avvisare l'operatore di introdurre dei dati.

Per introdurre ad esempio il valore del raggio di una circonferenza, che poi servirà al programma per calcolare l'area di un cerchio, si può scrivere l'istruzione **INPUT** in questo modo:

```
:
10 INPUT "INTRODUCI RAGGIO:" ; R
:
```

Quando il computer eseguirà l'istruzione 1Ø, sullo schermo apparità la scritta:

INTRODUCI RAGGIO:?

A questo punto l'operatore inizierà a battere il valore numerico del raggio di un cerchio, che sarà il contenuto della costante numerica di nome R.

Come si nota nell'istruzione 1Ø, tra l'espressione stringa ed il nome R c'è il carattere ; (punto e virgola). Questo carattere **deve essere sempre inserito tra la stringa ed il nome del dato da introdurre.**

L'istruzione INPUT con "messaggio"

Un'altra istruzione **INPUT** valida è la seguente:

```
3Ø INPUT "INTRODURRE DUE VALORI:" ; A,B
```

In questo caso sono da introdurre due valori numerici di nome A e B rispettivamente. Sullo schermo apparirà la scritta:

INTRODURRE DUE VALORI:?

In merito all'istruzione ora vista si può far notare che, se i dati da fornire sono piú di uno, essi vengono introdotti uno di seguito all'altro separati da una virgola.

Se in un'istruzione **INPUT** viene fornito un dato che non è in accordo con la variabile corrispondente, per esempio una lettera dell'alfabeto al posto di un numero, sullo schermo compare la scritta:

? REDO FROM START
(rifare dall'inizio)

la quale significa appunto che bisogna ribattere esattamente il dato.

L'istruzione **INPUT** è un'istruzione di tipo particolare, che a volte è difficoltoso inserire senza dover ricorrere a soluzioni drastiche (come spegnere il computer, cancellare il programma) o ad un arresto forzoso del programma stesso.

Si può notare, specialmente nei sistemi personal computer di piccola capacità, che qualsiasi dato si immetta con la tastiera, l'istruzione **INPUT** funziona quasi sempre, cioè accetta qualsiasi carattere: come poi questo carattere venga interpretato è un fatto veramente imprevedibile.

A volte, e questo succede quando l'operatore non sa quale tipo di valore deve introdurre dall'**INPUT**, il programma non esce perché il sistema automaticamente ricircola sulla stessa istruzione, finché non si introduce correttamente il dato (o i dati).

Per evitare inutili perdite di tempo o eventuali guasti al programma, bisogna uscire dall'**INPUT** eseguendo con la tastiera un'operazione che interrompa l'impasse.

Comunemente questa operazione consiste nel premere il tasto **CTRL** ed uno dei tanti caratteri presenti sulla tastiera.

Come sempre, le diverse modalità operative per interrompere un'istruzione **INPUT** dipendono dal tipo di sistema personal computer.

Successivamente, quando si analizzeranno piú a fondo le istruzioni BASIC, si imparerà ad eseguire diverse altre operazioni che, compiute tramite la tastiera, consentono di controllare dall'esterno una qualsiasi elaborazione.

A conclusione di quanto sinora detto in merito all'istruzione **INPUT**, si può aggiungere che, quando tale istruzione viene immessa nel computer senza numero di riferimento, sullo schermo appare il seguente messaggio:

```
? ILLEGAL DIRECT
  ERROR
```

il quale significa appunto che l'istruzione **INPUT** non può essere usata in modo diretto, ma soltanto nell'ambito di un programma.

4. – I primi comandi BASIC

Sui comandi BASIC, oltre a quanto precisato nelle lezioni precedenti, occorre aggiungere che molti di essi possono specificare ulteriori opzioni,

le quali consentono al sistema di operare soprattutto sulle varie unità in maniera piú completa ed efficace.

Alcuni sistemi personal computer consentono l'autonumerazione delle istruzioni con il seguente, apposito comando BASIC:

AUTO num1, num2

Con questo comando la numerazione delle istruzioni viene effettuata automaticamente dal sistema stesso a partire dal numero di riferimento num1 e con intervallo num2; per esempio, se si scrive:

AUTO 10, 10

il sistema prefigura sullo schermo il numero della prima istruzione, cioè:

10

Dopo il numero 1Ø l'operatore può iniziare a battere l'istruzione del programma.

Premendo il tasto **RETURN**, verrà automaticamente riprodotto il numero della seconda istruzione, ossia 2Ø.

Le successive istruzioni saranno numerate di dieci in dieci, ossia i loro numeri saranno: 3Ø, 4Ø e cosí via.

Il comando **LIST**, già trattato nelle precedenti lezioni, consente la visualizzazione delle istruzioni che costituiscono il programma presente nella memoria principale: scrivendo dopo la parola LIST appropriati numeri di riferimento, è possibile visualizzare una sola o piú istruzioni del programma.

Ad esempio per vedere sullo schermo il contenuto di programma compreso tra l'istruzione 1ØØ e l'istruzione 5ØØ, basta scrivere:

LIST 100-500

Per visualizzare la parte di programma che va dall'inizio all'istruzione 7Ø, è sufficiente scrivere:

LIST -70

Come già constatato, il comando **SAVE** consente la registrazione del programma presente in memoria su una unità di memoria ausiliaria (disco, nastro).

Dopo la parola SAVE possono essere scritte varie opzioni, ognuna con diversa funzione.

Per esempio, se si vuole dare un nome al programma che si intende registrare, si può scrivere:

SAVE "PROGRAMMA 1"

dove "PROGRAMMA 1" è il nome dato al programma da registrare.

Con alcuni personal computer è consentito assegnare un nome o una sigla anche al volume, cioè al supporto (disco o nastro); ad esempio, con il comando:

SAVE "VOL 1: PROG 2"

si registra il programma presente in memoria con il nome PROG 2 sul disco siglato VOL 1.

L'uso della "parola d'ordine" per la riservatezza dell'informazione

E' possibile addirittura scrivere l'opzione in una forma molto riservata, introducendo una parola d'ordine (in inglese, PASSWORD) dopo il nome del volume o del programma, come nel seguente caso:

SAVE "1: VOL 3: FILE 1/APRITI SESAMO"

dove il numero 1 dopo le virgolette sta ad indicare DRIVE n. 1 dell'unità a disco su cui avverrà la registrazione, VOL 3 è il nome del disco su cui si vuole registrare il programma di nome FILE 1 e di parola d'ordine (o chiave d'accesso):

APRITI SESAMO

Naturalmente la parola d'ordine è conosciuta soltanto da chi è "autorizzato" a leggere il contenuto del programma e non apparirà sullo schermo.

Come già verificato in precedenza, il comando **LOAD** si può definire l'operazione inversa del comando **SAVE**; esso consente cioè di caricare un programma da un'unità di memoria ausiliaria nella memoria principale del computer.

L'immissione avviene con le stesse opzioni con cui il programma è stato registrato.

Il comando **VERIFY** consente di "verificare" se la registrazione (effettuata con il comando **SAVE**) è stata eseguita in modo corretto; il sistema esegue la verifica controllando punto per punto le informazioni di programma presenti in memoria con quelle appena registrate.

Se dal controllo dei caratteri risulta che esiste differenza tra i due programmi, viene segnalato:

```
? VERIFY
  ERROR
```

A questo punto l'operatore deve provvedere a rieseguire il comando **SAVE**, eventualmente registrando il programma su un'altra zona del nastro o del disco e cambiando il nome del programma oppure il supporto (disco o nastro).

Una cattiva registrazione può dipendere da varie cause, una delle quali è l'usura dei supporti di memoria ausiliaria che devono sempre essere conservati con estrema cura negli appositi contenitori.

L'istruzione STOP (arresto temporaneo)

Prendiamo ora in considerazione l'istruzione **STOP**, che è strettamente legata al comando **CONT**. Essa serve ad arrestare l'elaborazione in un determinato punto del programma, per cui, se si vuole fare ripartire il programma dall'istruzione successiva, è sufficiente eseguire con la tastiera il comando BASIC:

CONT

Il comando CONT (per... continuare)

che significa appunto "Continua",

seguito da **RETURN** .

Per esempio, il seguente semplice programma:

```
10 PRINT "APRITI"
20 STOP
30 PRINT "SESAMO"
40 PRINT
45 PRINT
50 GO TO 10
```

quando viene eseguito con la **RUN**, visualizza la parola "APRITI" sullo schermo e poi si arresta con il messaggio:

BREAK IN 20

che significa appunto "arresto all'istruzione 20" dove c'è uno STOP.

Se si esegue il comando **CONT**, il computer continua ad elaborare a partire dall'istruzione successiva a quella di **STOP**, cioè dall'istruzione 30 **PRINT** "SESAMO".

Il comando **CONT** usato piú volte in un programma può servire per un controllo passo per passo dello stesso, quando si ha la necessità di controllare risultati parziali di una elaborazione complessa.

Naturalmente occorre sistemare nel programma stesso opportune stampe (PRINT) seguite da **STOP**, in modo che il programmatore possa avere il tempo di visualizzare i risultati parziali e decidere se continuare o interrompere il calcolo.

In modo diverso dall'istruzione **STOP** agisce l'altra istruzione di arresto precedentemente vista e denominata **END**. Quest'ultima arresta l'elaborazione nel punto stesso del programma in cui si trova; pertanto, se si vuole rieseguire il calcolo dall'inizio, occorre battere il comando **RUN**.

5. – Il controllo del programma

Quando l'elaborazione di un programma è avviata, il sistema personal computer consente all'operatore di intervenire, attraverso l'unità di input (tastiera), sul programma stesso interrompendolo in qualsiasi momento.

Per interrompere un'elaborazione

L'interruzione di una elaborazione si può effettuare in tanti modi, a seconda del particolare sistema personal computer che si sta usando.

Infatti i tasti utilizzati per interrompere un programma variano da sistema a sistema ed a volte, pur avendo la stessa funzione, la loro denominazione è diversa.

Ad esempio lo stesso tasto **RETURN**, che in alcuni personal computer serve per immettere un'istruzione o comando con la tastiera nell'unità

centrale del computer, in altri sistemi può essere denominato: **ENTER**, **CR**, **RET**, **RETN**, **CARRIAGE RETURN**.

In genere, si può dire che tutte le funzioni operative, come l'avvio, l'arresto, la continuazione di una elaborazione, l'immissione e la correzione di un programma, la cancellazione e l'inserimento di istruzioni, lo spostamento del cursore sullo schermo, ecc., variano in funzione del particolare sistema che si sta utilizzando. Tutto ciò si può spiegare con il fatto che le funzioni di controllo del programma esulano dalle regole proprie del linguaggio di programmazione: quest'ultimo è sempre indipendente dal sistema di elaborazione usato e può essere facilmente adattato a diversi computer, apportando lievi modifiche al programma; le funzioni di controllo, invece, sono legate alla particolare struttura fisica del computer.

Quindi, per utilizzare un sistema personal computer occorre studiare a fondo, sul relativo manuale d'uso, le relative funzioni operative di controllo, le quali sono valide soltanto per quel particolare sistema.

E' importante che ogni operazione di controllo **venga eseguita correttamente**, perché in caso contrario il programma può andare facilmente perso.

Una buona regola da seguire è quella di programmare nel programma stesso delle interruzioni con l'istruzione **STOP** e poi riprendere ogni volta l'elaborazione con il comando **CONT**.

L'operazione di interruzione

La maggior parte dei microsistemi utilizza però dei tasti funzionali esterni per interrompere il calcolo: ad esempio il tasto **CTRL** e, contemporaneamente, un tasto letterale come X o C.

Con questo sistema il calcolo si arresta e, con un messaggio, viene segnalato in quale punto del programma ciò è avvenuto.

Al momento dell'arresto si potrebbero anche eseguire delle verifiche sul programma stesso, stampando con un'istruzione **PRINT** immediata i valori di alcuni risultati parziali e dedurre da questi valori se un certo procedimento di calcolo sta proseguendo in modo corretto oppure se, prevedibilmente, non convergerà mai alla soluzione prevista.

Se l'operazione di interruzione è stata compiuta correttamente, è possibile riprendere l'elaborazione eseguendo in modo immediato un'istru-

zione di salto, come quella di seguito riportata:

GO TO num. oppure **GOTO** num.

dove num. è il numero dell'istruzione successiva a quella in cui si è verificato l'arresto.

Questa procedura non sempre è valida, soprattutto quando l'interruzione avviene durante un'istruzione ciclica come la **FOR... NEXT**.

Alcuni sistemi personal computer, come il VIC-20 Commodore, usano il tasto **RUN | STOP** abbinato con il tasto **RESTORE** per interrompere un'elaborazione; in tal modo però bisogna rieseguire l'elaborazione dal principio utilizzando il comando **RUN**, perché tutti i valori delle variabili ed i risultati parziali vengono ripristinati a Ø (zero) e soltanto il programma resta memorizzato in memoria.

In altri sistemi, premendo il tasto **CTRL** e contemporaneamente il tasto **RESET**, si ha la possibilità di azzerare tutta la memoria disponibile per l'utente, cioè programmi e dati, come se si fosse eseguito il comando **NEW**.

A proposito di azzeramento, quando si esegue il comando esecutivo **RUN** il sistema personal computer azzera tutti i contenuti delle variabili, tranne il programma, e compie l'elaborazione a partire dalla prima istruzione di programma.

Alcuni sistemi consentono l'utilizzo del comando **RUN** seguito da un numero di istruzione; in questo caso l'elaborazione inizierà da quel punto del programma.

6. - Il controllo del cursore e dello schermo

Nel linguaggio BASIC e nella maggior parte dei personal computer il cursore viene controllato esternamente da particolari tasti operativi, generalmente posizionati nella parte destra in basso della tastiera.

Questi tasti sono raffigurati con frecce che, a seconda della loro direzione, indicano i quattro movimenti che possono essere forniti al "cursore" o

carattere pronto, ossia in alto, in basso, a destra, a sinistra:

⇑ CRSR ⇓ ⇐ CRSR ⇒

Dall'interno di un programma i movimenti del cursore possono essere programmati attraverso particolari istruzioni di posizionamento (**TAB, HTAB, VTAB**), che verranno trattate piú diffusamente nella prossima lezione.

Per ora interessa conoscere, eseguendo appropriate operazioni con la tastiera, come posizionare il cursore in particolari punti del programma (listato che appare sullo schermo) al fine di eseguire inserimenti, apportare correzioni, cancellare qualche carattere, ecc.

Questa procedura, chiamata anche "editing", varia da sistema a sistema; in generale, comunque, si può dire che prima di posizionare il cursore occorre visualizzare sullo schermo l'istruzione o i gruppi di istruzioni sulle quali si devono eseguire operazioni di "editing", cioè di correzione.

Fatto ciò, si posiziona il cursore sul carattere da correggere, premendo ripetutamente l'opportuno tasto direzionale fino a che non appare il cursore e questo va a sovrapporsi al carattere stesso in cui si deve apportare la correzione.

Si noti che durante lo spostamento del cursore verso destra i caratteri che via via vengono sorpassati non vengono cancellati ma rispediti nella memoria, come se fossero stati ribattuti.

Operazioni di correzione al programma con l'uso del cursore

Si supponga per esempio di voler apportare delle correzioni alla parola "APRITI SESAMO" nell'istruzione errata:

```
10 PRINT "ABRITI SISAO"
```

Come si vede, occorre sostituire la lettera B con la lettera P nella parola ABRITI; la lettera I della parola SISAO con la lettera E, ed infine inserire la consonante M nella stessa parola.

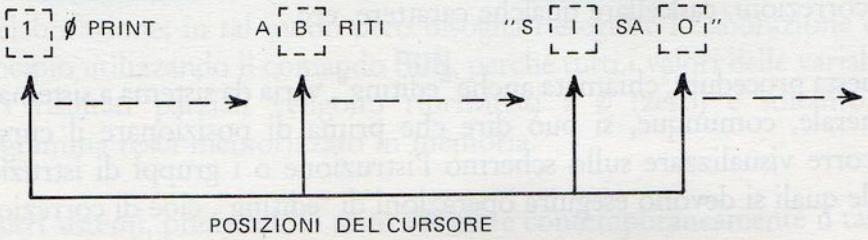
Dopo aver listato l'istruzione 10 con il comando **LIST 10**, bisogna posizionare il cursore sulla lettera B (errata).

Vi sono diversi criteri per posizionare il cursore su un determinato carattere di una istruzione: si può pigiare tante volte il tasto ⇐ CRSR ⇒ oppure (per alcuni sistemi) premere il tasto ESC e subito dopo il tasto

letterale **A**, il quale, ogni volta che viene premuto, fa muovere il cursore di una posizione a destra (per spostamenti in alto, in basso, e a sinistra si usano rispettivamente i tasti **B**, **C**, **D**).

In altri sistemi il posizionamento su un determinato carattere può essere attuato utilizzando un apposito programma: l'EDITOR, che consente di eseguire tra l'altro rapide e precise correzioni nel testo del programma.

Effettuato il posizionamento sul carattere B errato, si può battere il carattere corretto P e continuare a spostare il cursore sulla lettera I errata, come indicato di seguito.



POSIZIONI DEL CURSORE

Dopo che si è battuto il carattere corretto E (al posto della lettera errata I), si deve inserire la consonante M prima dell'ultima lettera O della parola SISAO. Anche per effettuare inserimenti, i criteri cambiano a seconda del tipo di personal computer.

Il tasto per l'inserimento di nuovi caratteri

Con alcuni sistemi, dopo aver posizionato il cursore sulla lettera "O", si batte il tasto speciale **INST** utilizzato appositamente per inserimenti; con altri sistemi si usano il tasto funzionale **CTRL** e contemporaneamente il tasto letterale **I** (che vuol dire appunto inserimento).

Dopo aver effettuato una di queste operazioni preliminari, si può iniziare ad inserire la lettera o i gruppi di lettere che mancano (nel nostro caso la consonante M); ad ogni carattere inserito corrisponde uno spostamento a destra della parte rimanente dell'istruzione.

Effettuato l'inserimento, si prosegue a spostare il cursore verso destra fino ad attraversare (per così dire) l'intera istruzione ed infine si batte il tasto **RETURN** per far sí che l'istruzione corretta venga immessa nella memoria del computer.

Il tasto per cancellare caratteri

Oltre a queste operazioni ne esistono numerose altre che consentono di modificare, ampliare, correggere in poco tempo intere parti di programma: per esempio, la cancellazione di caratteri viene effettuata tramite tasti speciali come il **DEL** (che sta per DELETARE cioè cancellare) oppure il

tasto ← (freccia rivolta a sinistra), il quale ogni volta che viene premuto cancella un singolo carattere posto alla sinistra del cursore e sposta di una posizione a sinistra tutta la rimanente parte destra dell'istruzione.

La cancellazione di intere istruzioni o gruppi di istruzioni, la rinumerazione delle stesse o dell'intero programma, l'autonumerazione, la riallocazione sono altri esempi di operazioni di "editing" che possono essere effettuate dai sistemi personal computer di un certo livello.

Infine, un'operazione di particolare importanza è quella per "schiarire" lo schermo e posizionare il cursore nell'angolo in alto a sinistra dello stesso.

Questa operazione è molto utile ed è paragonabile alla cancellazione di una lavagna dopo che questa diventa praticamente illeggibile.

Per "pulire" lo schermo TV

Per cancellare lo schermo (si noti che il programma è sempre in memoria) alcuni sistemi usano un tasto apposito, denominato CLEAR (che vuol dire appunto "schiarire", "cancellare"); altri sistemi impiegano un vero comando BASIC, cioè il comando:

CLEAR oppure **CLR**

che può essere ad esecuzione immediata o differita, ecc.

Il tasto HOME (a casa!)

Per il posizionamento del cursore nell'angolo in alto a sinistra molti sistemi usano il tasto apposito HOME ; altri utilizzano invece l'istruzione (immediata o differita):

HOME

Il motivo per cui per queste operazioni vengono usate apposite istruzioni BASIC risiede nel fatto che sia lo "schiarimento" dello schermo sia il posizionamento del cursore in alto a sinistra possono essere programmati dall'interno di un programma e consentire cosí l'esecuzione di grafici sullo schermo stesso, come se si trattasse di una vera lavagna.

Come è stato precisato nelle precedenti lezioni, lo schermo funge da "veicolo" di comunicazione visiva tra il programmatore ed il computer; quindi, è sempre utile visualizzare tutto ciò che viene comunicato dall'utente al computer e viceversa, in modo che l'elaborazione possa essere seguita "costruttivamente", senza dover perdere troppo tempo per la ricerca degli errori.

In seguito...

Nella prossima lezione verranno esaminati le costanti e le variabili (numeriche ed alfanumeriche), le espressioni algebriche, i comandi di utilità, le istruzioni ed i comandi per la raffigurazione grafica e del colore.

* * * * * * *

Da memorizzare

- Ogni istruzione in linguaggio BASIC è composta da tre parti: da un **numero** di riferimento, da una **parola chiave** o riservata (una parola inglese) e da una **espressione** algebrica o alfanumerica, che si definisce anche "oggetto di elaborazione".

- La programmazione strutturata, che è un insieme di tecniche compatte, consente di eliminare nel programma i salti e le deviazioni che possono far perdere al programmatore il filo logico nella sequenza di elaborazione.

- L'istruzione **REM** ha la funzione di descrivere il programma, ossia di **commentarlo**; essa non ha alcun valore operativo.

- La lunghezza di un'istruzione varia da computer a computer ed è compresa fra **80 caratteri e 256 caratteri**, inclusi il numero di riferimento, la parola chiave e gli spazi.

- Per eliminare un'istruzione da un programma è sufficiente battere il relativo numero di riferimento e premere il tasto **RETURN** .

- Alcuni tipi di personal computer utilizzano il comando BASIC **DELETE** oppure **DEL** per cancellare una o piú righe del programma.

- Tutto il complesso di regole, comandi o istruzioni, operazioni da tastiera che consente di introdurre il programma, listarlo, correggerlo, registrarlo, ecc., viene denominato **editing**.

- Nel linguaggio BASIC le istruzioni **eseguono operazioni logico-aritmetiche** su numeri e parole, mentre i comandi **istruiscono il personal computer** circa la gestione di tutte le unità del sistema.

- Il comando **NEW** consente di cancellare tutto quanto è presente nella memoria principale del computer.

- L'istruzione **PRINT**, che serve per visualizzare sullo schermo il contenuto di una certa variabile, quando è seguita dal simbolo # (diesis) consente di trasferire la visualizzazione o stampa ad una unità periferica stampante.

- L'istruzione **PRINT USING** (stampa personalizzata) permette di ottenere la rappresentazione di numeri o espressioni con spaziature diverse da quella standard.

- Il carattere , (virgola) contenuto nell'espressione oggetto dell'istruzione **PRINT** serve a spaziare un valore da un altro di un certo numero fisso di spazi dipendente dal tipo di personal computer.

- L'istruzione **INPUT** effettua l'ingresso dei dati dalla tastiera durante l'esecuzione del programma, dati che vengono depositati nella memoria centrale del computer. Quando l'istruzione **INPUT** è seguita dal simbolo #, viene effettuato il dialogo fra l'unità centrale di elaborazione del computer e la relativa memoria di massa.

- Se l'istruzione **INPUT** viene eseguita in modo diretto, sullo schermo appare il messaggio d'errore ILLEGAL DIRECT, il quale indica appunto che tale istruzione può essere usata soltanto nell'ambito di un programma.

- Il comando BASIC **AUTO** permette di ottenere la numerazione automatica delle righe del programma.

- Il comando BASIC **LIST** consente la visualizzazione sullo schermo delle istruzioni costituenti il programma presente nella memoria principale del computer.
- Il comando BASIC **SAVE** permette la registrazione del programma presente nella memoria principale su un'unità di memoria ausiliaria (disco o nastro magnetico).
- Il comando BASIC **LOAD** si può definire l'inverso del comando **SAVE**; esso consente cioè di caricare un programma da un'unità di memoria ausiliaria nella memoria principale.
- Il comando BASIC **VERIFY** consente di verificare se la registrazione (effettuata con il comando **SAVE**) è stata eseguita correttamente.
- L'istruzione **STOP** serve ad arrestare momentaneamente l'elaborazione in un determinato punto del programma; per fare ripartire il programma dall'istruzione successiva a quella di **STOP**, occorre eseguire il comando BASIC **CONT** (continuare) ad esecuzione immediata.
- L'istruzione **END** indica la fine del programma.
- Il tasto **RETURN**, che occorre premere al termine di ogni istruzione, viene anche denominato nei seguenti modi: **ENTER**, **CR**, **RET**, **RETN**, **CARRIAGE RETURN**.
- In genere, tutte le funzioni operative come l'avvio, l'arresto, la continuazione di un'elaborazione, l'immissione e la correzione di un programma, ecc., **variano in funzione del particolare sistema personal computer usato**.
- Il controllo dello spostamento del cursore sullo schermo è ottenuto con i tasti:

  ⇑ CRSR ⇓ e **⇐ CRSR ⇒**

  Agendo su tali tasti, è possibile posizionare il cursore in particolari punti del programma (listato sullo schermo) al fine di eseguire degli inserimenti, apportare delle correzioni, cancellare caratteri, ecc.
- Il comando BASIC **CLEAR** consente di cancellare lo schermo (mantenendo però sempre il programma nella memoria principale).
- L'istruzione **HOME** permette di posizionare il cursore nell'angolo in alto a sinistra dello schermo.

Scuola Radio Elettra
Torino

Pratica del BASIC

Lezione 4

1 — Come correggere gli errori in un programma
2 — Esercitazioni con l'istruzione **PRINT**
3 — Esercizi di programmazione

Che cosa farà

In questa lezione, tramite appositi esempi pratici, avrà modo di apprendere le diverse tecniche adottate per la correzione degli errori che possono essere commessi battendo un'istruzione.

Inoltre esaminerà dettagliatamente l'uso dei caratteri virgola (,) e punto e virgola (;) nell'istruzione PRINT, in modo da acquisire sicurezza sul loro impiego nella stesura di un programma.

Eseguirà infine alcuni interessanti esercizi di programmazione.

1. - Come correggere gli errori in un programma

In primo luogo si accerti che l'alimentatore sia regolarmente collegato al microcomputer e che questo sia connesso alla presa d'antenna del televisore. Accenda il microcomputer ed il televisore: sullo schermo deve apparire la consueta figura di presentazione standard.

Batta sulla tastiera del microcomputer l'istruzione immediata riportata sotto, nella quale è stato inserito appositamente un errore formale di sintassi, consistente nell'omissione delle due virgolette di apertura (") della stringa di caratteri:

LET A$ = SCUOLA RADIO ELETTRA"

e poi prema il tasto **RETURN**.

Sullo schermo vedrà apparire il seguente messaggio d'errore:

```
? TYPE MISMATCH ERROR
READY
```

Questo messaggio significa mancata corrispondenza di tipo di variabile (nel nostro caso, variabile alfanumerica).

Per inserire le virgolette nell'istruzione battuta prima, si può seguire la

procedura descritta nella lezione *Analisi e Programmazione 4*, consistente nell'utilizzare i movimenti del cursore, il quale può essere spostato in qualsiasi posizione sullo schermo manovrando opportunamente i seguenti tasti situati nella parte inferiore destra della tastiera:

SHIFT ⇑ CRSR ⇓ ⇐ CRSR ⇒

Come prima operazione occorre portare il cursore sulla lettera L della parola chiave **LET**. Per fare ciò prema il tasto SHIFT e, tenendolo abbassato, batta il tasto ⇑ CRSR ⇓ tante volte quante sono necessarie per portare appunto il cursore sulla lettera L della parola **LET**.

Come può notare, questa operazione, spiegata in altri termini, consiste nello spostare verso l'alto il cursore.

Posizionato il cursore sulla lettera L della parola LET, occorre spostarlo verso destra e posizionarlo sulla lettera S di SCUOLA.

Per effettuare questa operazione si deve usare il tasto ⇐ CRSR ⇒ il quale, ogni volta che viene battuto, fa spostare il cursore di una posizione a destra. Batta perciò il tasto ⇐ CRSR ⇒ tante volte quante sono necessarie affinché il cursore vada a sovrapporsi alla lettera S della parola SCUOLA.

A questo punto si può procedere all'inserimento del carattere virgolette (").

Prema prima il tasto SHIFT e, tenendolo abbassato, prema una sola volta il tasto INST | DEL posto nell'angolo superiore destro della tastiera.

Può subito notare che la frase letterale:

SCUOLA RADIO ELETTRA"

subisce uno spostamento laterale a destra di un carattere (gli spostamenti sono tanti quante le battiture del tasto INST | DEL , cioè quanti sono gli inserimenti da effettuare).

Dopo aver eseguito questa operazione, si può inserire il carattere (").

Prema quindi il tasto SHIFT e, tenendolo abbassato, batta il tasto " | 2 .

A questo punto l'istruzione è corretta e può essere immessa nella memoria centrale del computer: prema perciò il tasto RETURN .

Come può rilevare osservando lo schermo, il cursore è ora posizionato sulla lettera R della parola READY, per cui occorre spostarlo al di sotto della parola stessa.

Batta pertanto una volta il tasto ⇑ CRSR ⇓ , in modo da portare il cursore nella posizione sotto indicata:

READY

L'istruzione ad esecuzione immediata che è stata battuta, ossia:

LET A$ = "SCUOLA RADIO ELETTRA"

fa sí che la frase (stringa) "SCUOLA RADIO ELETTRA" sia memorizzata nella cella di memoria chiamata A$.

Per vedere (e quindi verificare) che cosa c'è in A$, è sufficiente eseguire la seguente istruzione:

PRINT A$

e al termine battere sempre il tasto RETURN .

Sullo schermo comparirà la scritta:

SCUOLA RADIO ELETTRA

Se invece si battesse erroneamente l'istruzione:

PRINT A

premendo il tasto RETURN si otterrebbe sullo schermo:

Ø
READY

ossia zero, perché la cella A è diversa dalla cella A$ contenente la frase vista prima; inoltre, il nome A può essere usato per "nominare" **variabili numeriche e non stringhe alfanumeriche.**

Qualora si accorgesse di aver commesso un errore mentre ancora sta

battendo un'istruzione (ossia quando non ha ancora premuto il tasto **RETURN**), può immediatamente correggere l'errore stesso, come già accennato nella precedente lezione, premendo il tasto **INST | DEL** , il quale consente di fare scomparire il carattere che si trova immediatamente a sinistra del cursore.

Batta, per esempio, la seguente istruzione immediata, inserendo volutamente la lettera Q al posto della lettera A nella parola SCUOLA:

LET A$ = "SCUOLQ

Per cancellare l'errore, batta una volta il tasto **INST | DEL** , in modo da portare il cursore sulla lettera Q posta alla fine della parola SCUOLQ, quindi batta la lettera corretta A.

Come già precedentemente precisato, occorre infine tenere sempre ben presente che in un programma è possibile ribattere in qualsiasi momento un'istruzione errata: il computer sostituirà automaticamente la precedente istruzione con quella nuova.

E' possibile inoltre cancellare un'istruzione indesiderata battendo semplicemente il numero di quella istruzione e premendo il tasto **RETURN** : l'istruzione completa verrà cancellata dalla memoria principale del computer.

Per verificare che le correzioni apportate alle istruzioni di un programma siano state regolarmente immesse, è sempre opportuno battere il comando **LIST** per visualizzare sullo schermo il programma stesso corretto.

2. - Esercitazioni con l'istruzione **PRINT**

Questa istruzione, che può essere immediata oppure differita, è stata piú volte utilizzata nelle precedenti lezioni; come si è visto, essa serve a visualizzare sullo schermo il contenuto di una o piú variabili.

Vedremo ora quale funzione svolgono la virgola (,) ed il punto e virgola (;) che separano le variabili dell'istruzione in questione.

Prima di procedere alle esercitazioni vere e proprie, è però opportuno ricordare che il Suo microcomputer, per la rappresentazione sullo scher-

mo dei caratteri (lettere, simboli, ecc.), dispone di 1.000 posizioni di stampa, suddivise in 25 righe e 40 colonne, come schematizzato nella *fig. 1.*

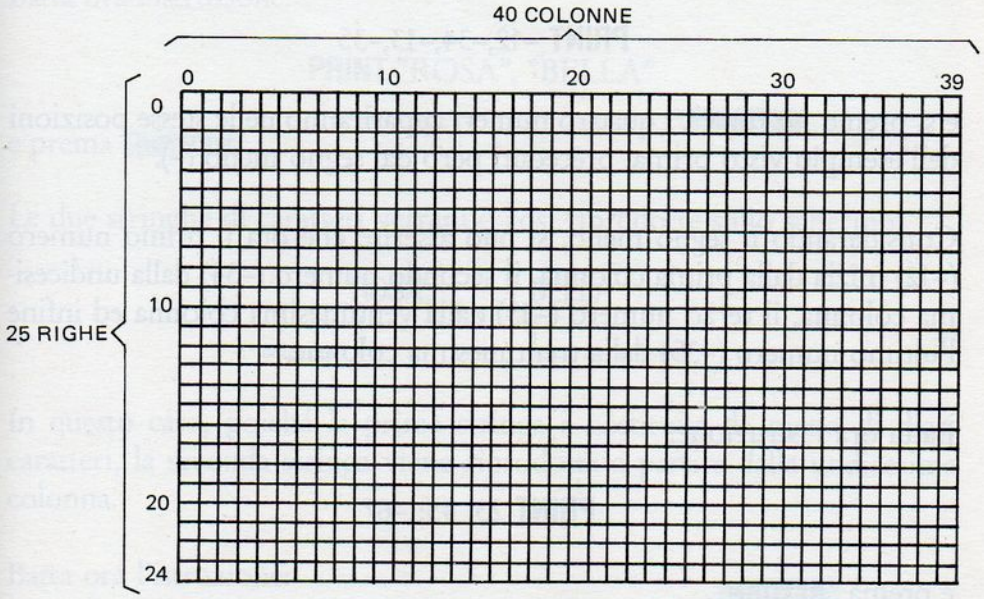


Fig. 1 – Posizioni di stampa dei caratteri sullo schermo del computer.

Ciò premesso, cancelli lo schermo, cioè prema il tasto **SHIFT** e, mantenendolo premuto, abbassi il tasto **CLR | HOME** .

Batta ora sulla tastiera la seguente istruzione:

PRINT 12,34,13,35

e prema il tasto **RETURN** .

Sullo schermo i due numeri verranno cosí riprodotti:

12 34 13 35

2ª colonna — 12ª colonna — 22ª colonna — 32ª colonna

Il numero 12 cioè apparirà sul lato sinistro dello schermo a partire dalla seconda colonna, il numero 34 a partire dalla dodicesima colonna, il numero 13 a partire dalla ventiduesima colonna ed il numero 35 dalla trentaduesima colonna.

Il primo numero inizia dalla seconda colonna anziché dalla prima per il

fatto che si deve permettere la stampa del segno meno (-) nel caso in cui il numero sia negativo.

Infatti, se si batte l'istruzione:

PRINT -12,-34,-13,-35

e si preme **RETURN**, i quattro numeri appariranno nelle stesse posizioni dell'esempio visto prima, preceduti però dal segno meno (-).

Considerando il segno meno, si può asserire che ora il primo numero (-12) inizia dalla prima colonna, il secondo numero (-34) dalla undicesima colonna, il terzo numero (-13) dalla ventunesima colonna ed infine l'ultimo numero (-35) dalla trentunesima colonna.

Batta ora l'istruzione:

PRINT 23; 45; -67

e prema **RETURN**.

Sullo schermo vedrà apparire:

23 45 -67

Come potrà rilevare, i tre numeri vengono riprodotti uno di seguito all'altro, ma, mentre i primi due numeri sono separati fra loro da due spazi, fra il secondo e il terzo numero si ha un solo spazio di separazione e poi il segno meno (-).

Da quanto sinora constatato, si può asserire quanto segue.

- Il separatore virgola provvede a dividere lo schermo in quattro aree di stampa, ciascuna delle quali costituita da dieci colonne.

  In altre parole, la virgola provvede a tabulare ogni numero nella successiva area disponibile.

  Se in una istruzione **PRINT** i numeri da stampare sono superiori a quattro, il quinto numero verrà stampato nella riga successiva. A questo riguardo occorre precisare che le aree di stampa dei numeri sullo schermo variano da computer a computer.

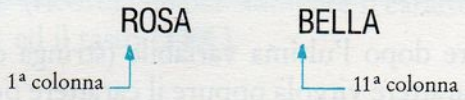
- Il separatore punto e virgola provvede invece a far stampare i numeri sullo schermo uno di seguito all'altro, facendoli precedere e seguire da due spazi.

Batta ora l'istruzione:

PRINT "ROSA", "BELLA"

e prema **RETURN** .

Le due stringhe di caratteri verranno cosí riprodotte sullo schermo:

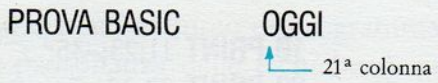


In questo caso, poiché la prima stringa è costituita da meno di dieci caratteri, la seconda stringa viene riprodotta a partire dalla undicesima colonna.

Batta ora l'istruzione:

PRINT "PROVA BASIC", "OGGI"

e prema **RETURN** ; sullo schermo apparirà la scritta cosí disposta:



Infatti, essendo la prima stringa costituita da dieci caratteri, non vi è piú spazio per iniziare la seconda stringa alla undicesima colonna (seconda area di stampa), per cui la seconda stringa viene riprodotta a partire dalla ventunesima colonna, ossia dalla terza area di stampa.

Cancelli ora lo schermo, premendo il tasto **SHIFT** ed il tasto **CLR | HOME** ; batta poi la seguente istruzione:

PRINT "ROSA"; "BELLA"

e prema **RETURN** ; vedrà comparire sullo schermo la scritta:

ROSABELLA

ossia le due stringhe saranno stampate senza alcuno spazio separatore.

Quanto finora visto può essere cosí sintetizzato.

- Se il separatore fra le due stringhe è costituito dalla virgola e la prima stringa è composta da meno di dieci caratteri, la seconda stringa viene stampata a partire dalla undicesima colonna (seconda area di stampa); se invece la prima stringa è formata da dieci o piú caratteri, la seconda stringa viene stampata a partire dalla ventunesima colonna (terza area di stampa).

- Se il separatore fra le stringhe è costituito dal punto e virgola, le stringhe vengono stampate una di seguito all'altra senza separazione.

E' possibile riportare dopo l'ultima variabile (stringa o numero) dell'istruzione **PRINT** il carattere virgola oppure il carattere punto e virgola. In questo caso un'eventuale, successiva istruzione **PRINT** inizierà a visualizzare i suoi valori sulla stessa riga della precedente istruzione **PRINT**, mantenendo però valido il significato del separatore usato; in altre parole, **il cursore non viene trasferito a capo.**

Il semplice esempio che segue servirà a chiarire ulteriormente quanto ora espresso.

Batta le seguenti istruzioni differite, premendo il tasto RETURN al termine di ciascuna di esse.

```
10 PRINT 11;23;-35;
20 PRINT 22;32;-7
30 PRINT 5,-6,8,
40 PRINT 2,12,13
50 PRINT "ROSSO";
60 PRINT "BLU",
70 PRINT "VERDE"
```

Esegua il programma battendo il comando BASIC **RUN** e premendo RETURN ; sullo schermo comparirà:

```
11  23 -35  22  32 -7
5         -6        8          2
12         13
ROSSOBLU VERDE
```

A conclusione di queste esercitazioni, occorre ancora precisare che la

parola riservata **PRINT** può essere scritta in modo abbreviato mediante il solo carattere ?, consentendo cosí di risparmiare tempo nella stesura di un programma.

Per verificare la veridicità di quanto asserito, cancelli lo schermo con il solito sistema.

Batta quindi sulla tastiera la seguente istruzione immediata:

? 12,34

e prema **RETURN** (ricordi che per stampare il carattere ? occorre premere il tasto **SHIFT** ed il tasto **? | /**).

Sullo schermo vedrà apparire:

12 34

Potrà cosí constatare che i due numeri sono visualizzati sullo schermo come se fosse stata battuta la parola riservata **PRINT**.

Può ora accingersi all'esecuzione dei previsti esercizi di programmazione.

Abbia cura di leggere sempre con la massima attenzione l'analisi del diagramma di flusso e la descrizione della relativa codifica in linguaggio BASIC che accompagnano ciascun esempio di programmazione riportato nelle lezioni. Infatti, la padronanza delle tecniche di programmazione non si acquista soltanto scrivendo e provando programmi, ma soprattutto cercando di comprendere a fondo quelli presentati nelle lezioni, i quali sono stati appositamente studiati per consentire di assimilare gradualmente le procedure di programmazione del computer.

3. – Esercizi di programmazione

1° ESERCIZIO

Proviamo a riprogrammare il calcolo della somma di N numeri, apportando al programma "PG2" della precedente lezione opportune modifiche, tali da rendere il calcolo stesso maggiormente flessibile ed utilizzabile piú volte.

Si tratta di sostituire il flow-chart riportato nella *fig. 1* della lezione *Pratica del BASIC 3* con un altro che consenta all'operatore di effettuare quante operazioni di somma o differenza egli desidera.

Invece del parametro N, che sta ad indicare quante operazioni di somma devono essere calcolate, è possibile introdurre il carattere S per comunicare al computer che si vuole continuare a calcolare oppure il carattere N per dirgli di troncare ad un certo punto l'elaborazione e di visualizzare successivamente il risultato della somma totalizzata fino a quel punto.

Inoltre, utilizzando l'istruzione di arresto temporaneo

STOP

ed eseguendo successivamente il comando BASIC **CONT**, si potrebbero eseguire a volontà tanti calcoli parziali quanti se ne desiderano, cioè si può rendere l'elaborazione ciclica.

Nella *fig. 2* è riportato il diagramma di flusso in questione, che ora analizzeremo.

Nel blocco 1) viene assegnato alla variabile di nome D il valore Ø (zero), che poi sarà assunto come risultato della somma.

Nel blocco 2) viene rappresentata un'istruzione di **INPUT** con visualizzazione di messaggio: "INTR.DATO O N"; tale messaggio deve rammentare all'operatore che egli può inserire o un valore numerico o la lettera N per terminare la somma algebrica.

Nel blocco 3) il programma si chiede se il dato appena introdotto è un nuovo addendo (cioè un valore numerico) oppure è una stringa letterale di un carattere (la lettera N che significa il non proseguimento del calcolo).

Se il valore appena introdotto risulta diverso dalla lettera N, l'elaborazione prosegue con il blocco 4), dove il dato numerico appena introdotto va ad accumularsi nella variabile di accumulo di nome D, e poi ricomincia a richiedere un altro dato.

Se invece il dato appena introdotto è proprio la lettera N, l'elaborazione prosegue per il blocco 5), dove viene visualizzato il risultato D della somma.

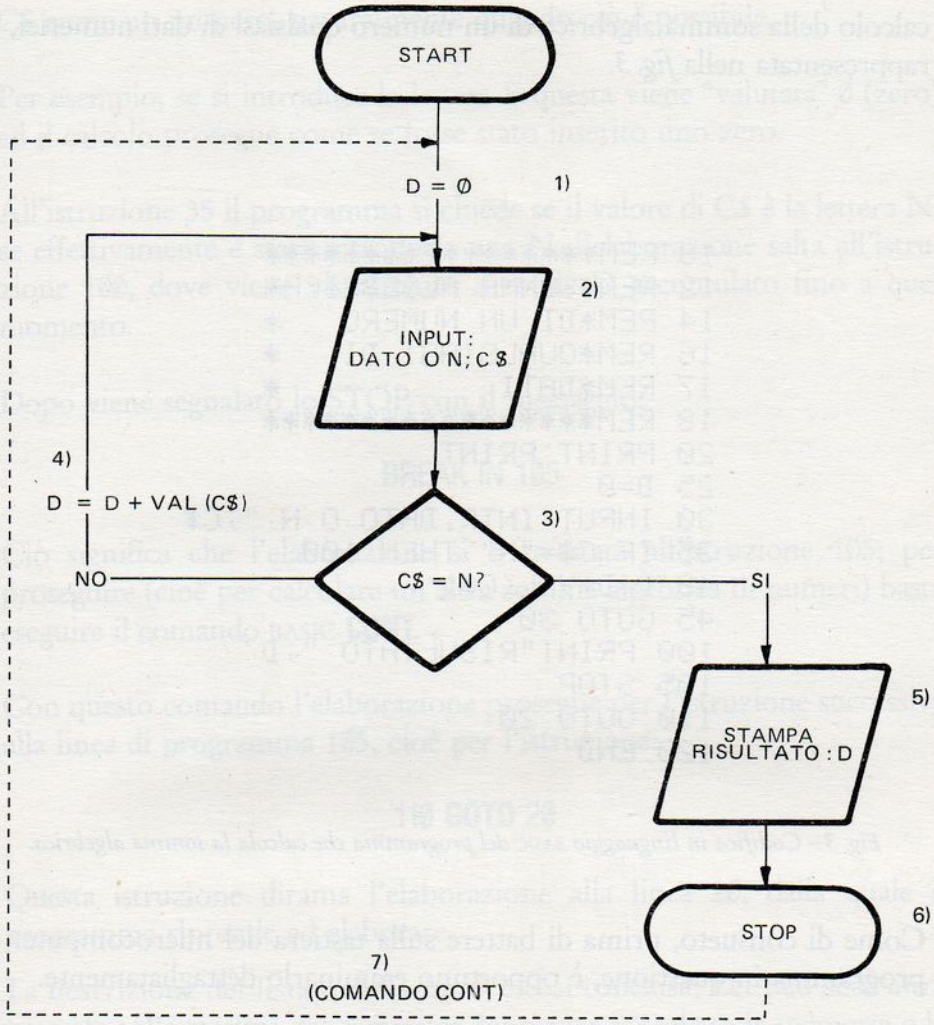


Fig. 2 – Flow-chart del calcolo della somma algebrica di un numero qualsiasi di dati numerici.

Dopo la visualizzazione del risultato viene eseguita un'istruzione **STOP** che, come è noto, consente di ottenere un arresto parziale dell'elaborazione.

Scrivendo un'istruzione di salto incondizionato al blocco 1), subito dopo l'istruzione STOP il programma ricomincerà ad eseguire il calcolo di un'altra somma algebrica.

Per attivare il proseguimento dell'elaborazione dopo l'arresto temporaneo, occorre eseguire il comando BASIC **CONT**.

La codifica in linguaggio BASIC del flow-chart ora descritto, relativo al

calcolo della somma algebrica di un numero qualsiasi di dati numerici, è rappresentata nella *fig. 3*.

```
10 REM*******************
12 REM*SOMMA ALGEBRICA*
14 REM*DI UN NUMERO    *
16 REM*QUALSIASI DI    *
17 REM*DATI            *
18 REM*******************
20 PRINT:PRINT
25 D=0
30 INPUT"INTR.DATO O N:";C$
35 IF C$="N" THEN 100
40 D=D+VAL(C$)
45 GOTO 30
100 PRINT"RISULTATO:";D
105 STOP
110 GOTO 20
120 END
```

Fig. 3 – Codifica in linguaggio BASIC *del programma che calcola la somma algebrica.*

Come di consueto, prima di battere sulla tastiera del microcomputer il programma in questione, è opportuno esaminarlo dettagliatamente.

Le istruzioni 10÷18 sono di commento al programma e non presentano particolarità degne di nota.

Nell'istruzione 25, come visto in precedenza, viene assegnato alla variabile D il valore Ø.

Nell'istruzione 30 c'è una **INPUT** con messaggio, il quale avverte l'operatore che può introdurre un qualsiasi dato numerico (con o senza segno) oppure la lettera N per troncare l'elaborazione.

Il valore del dato introdotto (numero o lettera N) viene chiamato C$, cioè è considerato un valore stringa che poi viene interpretato (nell'istruzione 40) come numero; infatti all'istruzione 40 si arriva soltanto se è stato introdotto un valore diverso dalla lettera N.

L'istruzione **VAL (C$)** consente al computer di interpretare il contenuto di

C$ come un numero, naturalmente quando ciò è possibile.

Per esempio, se si introduce la lettera F questa viene "valutata" Ø (zero) ed il calcolo prosegue come se fosse stato inserito uno zero.

All'istruzione 35 il programma si chiede se il valore di C$ è la lettera N: se effettivamente è stata introdotta una N, l'elaborazione salta all'istruzione 1ØØ, dove viene visualizzato il totale D accumulato fino a quel momento.

Dopo viene segnalato lo STOP con il messaggio:

BREAK IN 1Ø5

Ciò significa che l'elaborazione si è arrestata all'istruzione 1Ø5; per proseguire (cioè per calcolare un'altra somma algebrica di numeri) basta eseguire il comando BASIC **CONT**.

Con questo comando l'elaborazione prosegue per l'istruzione successiva alla linea di programma 1Ø5, cioè per l'istruzione:

11Ø GOTO 2Ø

Questa istruzione dirama l'elaborazione alla linea 2Ø, dalla quale il programma riprende ad elaborare.

La descrizione del listato della *fig. 3* è così conclusa; Lei può senz'altro batterlo sulla tastiera del computer dopo aver cancellato la memoria o lo schermo, tenendo presenti le avvertenze fornite al riguardo nelle precedenti lezioni e ricordando sempre di premere il tasto **RETURN** al termine di ogni linea di programma.

Terminata la battitura del programma, è opportuno listarlo sullo schermo con il comando BASIC:

LIST

seguito da **RETURN**.

Controlli con cura il listato che appare sullo schermo confrontandolo con quello riportato nella *fig. 3* e corregga gli eventuali errori.

Se tutto è regolare, può verificare il funzionamento del programma ora immesso nel computer.

Batta quindi il comando BASIC:

RUN

seguito da **RETURN** ; sullo schermo apparirà la scritta:

INTR. DATO O N : ?

A questo punto può via via introdurre nel computer i numeri, sia positivi sia negativi (questi ultimi preceduti dal segno meno), da sommare; ad ogni dato numerico che viene battuto sulla tastiera, sullo schermo comparirà la scritta:

INTR. DATO O N : ?

Per ottenere il risultato della somma è sufficiente battere la lettera N.

Supponga per esempio di volere sommare fra loro i seguenti tre numeri: 12Ø, –75 e 225; sullo schermo compariranno le seguenti scritte:

```
INTR. DATO O N :?  12Ø
INTR. DATO O N :? -75
INTR. DATO O N :?  225
INTR. DATO O N :? N
RISULTATO       :  27Ø
BREAK IN 1Ø5
```

Volendo eseguire altre somme, sarà sufficiente che batta sulla tastiera il seguente comando BASIC:

CONT

seguito da **RETURN** .

Anche questo programma, che denomineremo "PG3", può essere registrato sul lato B della cassetta-nastro ricevuta con la prima lezione, eseguendo il comando BASIC:

SAVE "PG3"

seguito da **RETURN** .

Per eseguire questa operazione occorre attenersi scrupolosamente alle istruzioni fornite al riguardo nella precedente lezione, al fine di posizionare la testina di lettura/scrittura del registratore subito dopo il programma "PG2", già registrato sul nastro magnetico.

Anziché con il nome "PG3" il programma in oggetto può essere denominato con un altro nome a piacere, purché sia diverso da quelli assegnati ai programmi già registrati sulla cassetta.

Se si registrasse un file-programma con lo stesso nome di un altro programma registrato, quest'ultimo verrebbe "inesorabilmente" cancellato sul nastro; inoltre, si perderebbero anche quelli registrati successivamente.

Nella *fig. 4* è riportata la configurazione del sistema di registrazione dei file-programmi sul nastro magnetico della cassetta. I gap di registrazione sono zone di nastro magnetico disposte fra due registrazioni successive, che non contengono alcuna informazione (zone "vuote"); essi si formano nella fase di avvio e di arresto del nastro magnetico, quando la sua velocità di scorrimento non è costante e quindi non è possibile effettuare registrazioni.

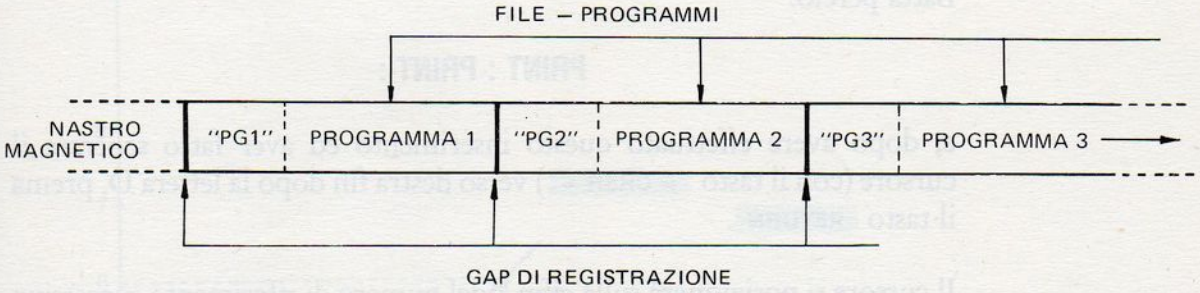


Fig. 4 – Registrazione dei file-programmi su cassetta nastro.

Dopo avere registrato (ossia salvato) il nuovo programma, occorre aggiornare l'elenco archivio dei file-programmi come descritto nella lezione precedente, in modo che il programma possa preliminarmente essere rintracciato a tavolino.

Se si vuole correggere e nuovamente registrare un programma precedentemente archiviato sul nastro, occorre reimmetterlo nella memoria principale del computer (eseguendo il comando BASIC **LOAD** "NOME DEL PROGRAMMA"), correggerlo, rieseguirlo con il comando **RUN** ed infine registrarlo con un nuovo nome in fondo alla lista dei programmi già registrati.

Se, per esempio, dopo avere registrato il programma "PG3" si desidera correggerlo o modificarlo in qualche punto, occorre reimmetterlo nella memoria principale (se già non c'è) e listare l'istruzione o il gruppo di istruzioni che si vuole correggere.

Supponga ora di dover correggere l'istruzione 1ØØ del programma in questione come di seguito indicato:

1ØØ PRINT : PRINT : PRINT "RISULTATO:"; D

Utilizzando i tasti **SHIFT** ed i tasti **⇑ CRSR ⇓** e **⇐ CRSR ⇒** , posizioni il cursore sulla cifra 1 del numero 1ØØ e successivamente sulla lettera P della parola chiave **PRINT**. Quindi esegua la funzione di inserimento.

Prema poi il tasto **SHIFT** e batta dodici volte il tasto **INST | DEL** , ossia tante volte quanti sono i caratteri che occorre inserire (nel nostro caso la parte da aggiungere PRINT : PRINT: è appunto costituita da dodici caratteri).

Dopo avere creato uno spazio di dodici caratteri, può inserire la parte di istruzione aggiuntiva.

Batta perciò:

PRINT : PRINT :

e, dopo avere effettuato questo inserimento ed aver fatto scorrere il cursore (con il tasto **⇐ CRSR ⇒**) verso destra fin dopo la lettera D, prema il tasto **RETURN** .

Il cursore si posizionerà sulla cifra 1 del numero di riferimento successivo (1Ø5).

A questo punto occorre portare il cursore in fondo alla lista del programma, al di sotto della parola READY.

Prema quindi il tasto **⇑ CRSR ⇓** sino a portare il cursore nella posizione indicata.

Può notare che la linea di programma 1ØØ, ora modificata, comprende tre istruzioni separate da : (due punti).

Le prime due **PRINT** servono a separare la visualizzazione del risultato (D) dalla stampa della precedente istruzione.

2° ESERCIZIO

Questo semplice esercizio di programmazione consiste nel calcolare e visualizzare i punti di una retta di equazione Y=X+1, dove X è un qualsiasi valore reale da introdurre mediante la tastiera del computer.

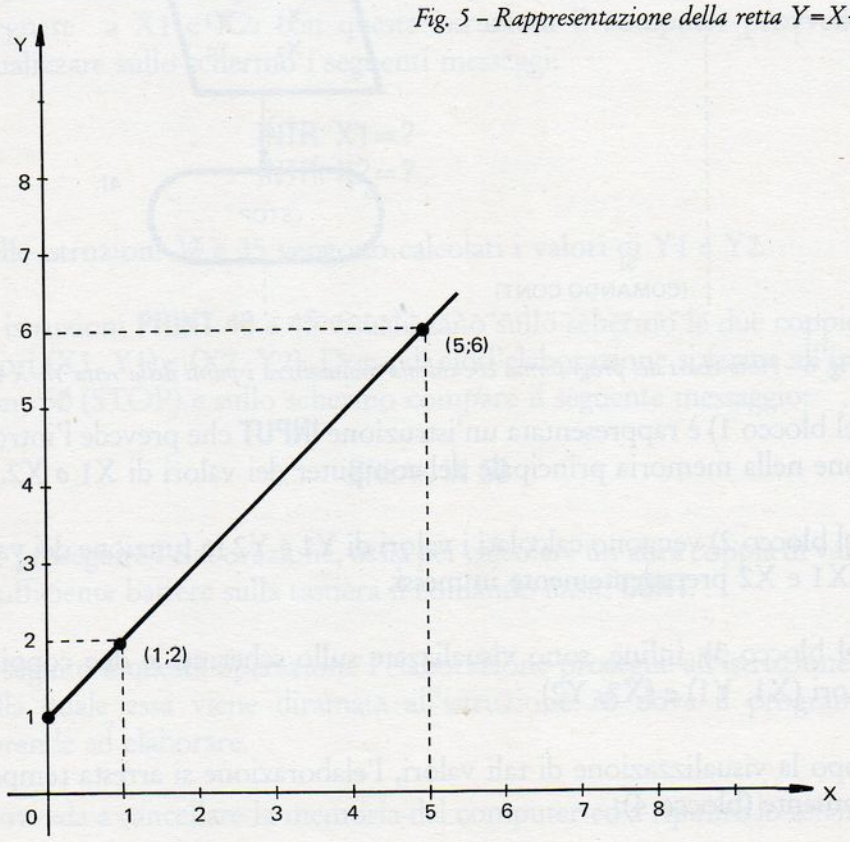
Come è noto, per tracciare una retta sono sufficienti due soli punti (ossia due soli valori di X).

A questo riguardo è opportuno fare una breve premessa per ricordare come viene rappresentata graficamente l'equazione della retta in questione.

Se si assegnano alla variabile X dei valori, Y assume a sua volta valori ben determinati; per esempio, se X=1, Y=1+1=2; se X=5, Y=5+1=6 e cosí via.

Ognuna delle coppie di valori cosí ottenute, riportate in un sistema di assi cartesiani, determina un punto e l'unione di questi punti definisce il grafico, il quale, come si vede chiaramente nella *fig. 5*, è una retta.

Fig. 5 – Rappresentazione della retta Y=X+1.



Il flow-chart relativo alla risoluzione del problema che visualizza i punti della retta Y=X+1 è riportato nella *fig. 6*.

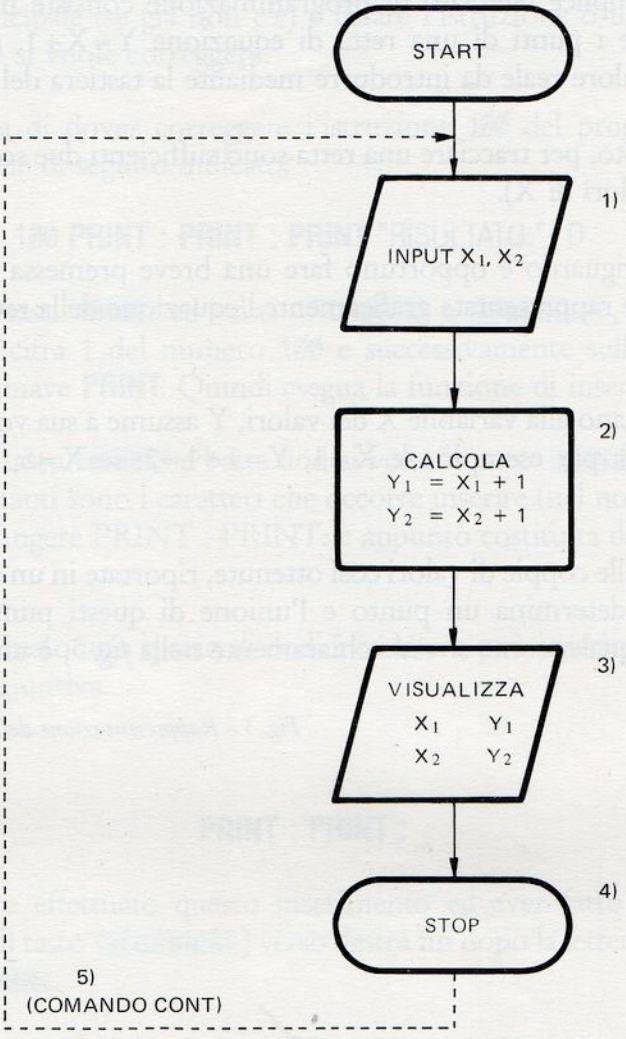


Fig. 6 – Flow-chart del programma che calcola e visualizza i punti della retta Y=X+1.

Nel blocco 1) è rappresentata un'istruzione **INPUT** che prevede l'introduzione nella memoria principale del computer dei valori di X1 e X2.

Nel blocco 2) vengono calcolati i valori di Y1 e Y2 in funzione dei valori di X1 e X2 precedentemente immessi.

Nel blocco 3), infine, sono visualizzate sullo schermo le due coppie di valori (X1, Y1) e (X2, Y2).

Dopo la visualizzazione di tali valori, l'elaborazione si arresta temporaneamente (blocco 4).

Volendo ottenere un'altra coppia di valori, si esegue il comando BASIC **CONT**, che realizza un salto incondizionato al blocco 1).

La codifica in linguaggio BASIC che calcola e visualizza i punti della retta Y=X+1 è mostrata nella *fig. 7*.

```
10 REM****************
12 REM* RETTA Y=X+1 *
15 REM****************
20 INPUT"INTR.X1=";X1
25 INPUT"INTR.X2=";X2
30 Y1=X1+1
35 Y2=X2+1
40 PRINT:PRINTX1,Y1
45 PRINT:PRINTX2,Y2
50 STOP
55 GOTO 20
60 END
```

Fig. 7 - Codifica in BASIC del programma "RETTA 1".

Le istruzioni 20 e 25 prevedono l'introduzione dei due dati numerici da assegnare a X1 e X2: con queste istruzioni il computer provvede a visualizzare sullo schermo i seguenti messaggi:

```
INTR. X1=?
INTR. X2=?
```

Nelle istruzioni 30 e 35 vengono calcolati i valori di Y1 e Y2.

Le istruzioni **PRINT** 40 e 45 visualizzano sullo schermo le due coppie di valori (X1, Y1) e (X2, Y2). Dopo di ciò l'elaborazione si ferma all'istruzione 50 (STOP) e sullo schermo compare il seguente messaggio:

```
BREAK IN 50
```

Per proseguire l'elaborazione, ossia per calcolare un'altra coppia di valori, è sufficiente battere sulla tastiera il comando BASIC **CONT**.

In seguito a questa operazione l'elaborazione prosegue all'istruzione 55, dalla quale essa viene diramata all'istruzione 20 dove il programma riprende ad elaborare.

Provveda a cancellare la memoria del computer ed a ripulire lo schermo

dalle scritte, quindi batta sulla tastiera il nuovo programma.

Dopo avere listato e controllato il programma immesso nel computer, confrontandolo con quello riportato nella *fig. 7*, ed aver eliminato eventuali errori, può passare ad eseguirlo battendo il comando BASIC **RUN** seguito da **RETURN** .

Vedrà apparire sullo schermo la scritta:

```
INTR.X1=?
```

Introduca pertanto il numero da attribuire alla variabile X1 (per esempio, facendo riferimento alla *fig. 5*, il numero 1) e prema **RETURN** .

Sullo schermo comparirà:

```
INTR.X2=?
```

Introduca ora il numero da attribuire alla variabile X2 (per esempio, facendo sempre riferimento alla *fig. 5*, il numero 5) e prema **RETURN** .

Sullo schermo appariranno le due coppie di valori:

```
1            2
5            6
BREAK IN 50
```

Tali coppie di valori corrispondono ai punti fra i quali si può tracciare la retta di equazione Y=X+1 (*fig. 5*).

Anche questo programma può essere registrato sulla cassetta in coda agli altri programmi precedentemente registrati ed il nome da assegnargli può essere "RETTA 1". Ricordi di seguire la procedura di registrazione descritta nella lezione *Pratica del BASIC 3*; inoltre scriva il nome di quest'ultimo programma nell'elenco programmi presentato nella stessa lezione pratica.

Terminata la registrazione del programma, cancelli sia la memoria del computer, con il comando **NEW** seguito da **RETURN**, sia lo schermo premendo i tasti **SHIFT** e **CLR | HOME** .

Come ha potuto constatare mediante gli esercizi ora eseguiti, l'istruzione **STOP** consente di interrompere l'esecuzione dell'elaborazione.

In un programma possono essere impiegate piú istruzioni di **STOP**.

In particolare, questa istruzione si rivela molto utile quando in un programma vi sono diverse diramazioni e si vuole porre fine all'elaborazione al termine di ciascuna di esse.

3° ESERCIZIO

Questo esercizio di programmazione, che effettua il confronto fra due numeri, illustra appunto l'uso di piú istruzioni **STOP** nello stesso programma.

Il relativo flow-chart, che comprende tre diramazioni terminanti con l'istruzione **STOP**, è riportato nella *fig. 8*.

Nel blocco 1) sono rappresentate due istruzioni di **INPUT**, che prevedono l'immissione nel computer dei valori dei numeri A e B da confrontare tra loro.

Nel blocco 2) il programma si chiede se A<=B (ossia se il numero A è inferiore oppure uguale al numero B); in caso negativo (ossia se il numero A è maggiore del numero B) l'elaborazione prosegue per il blocco 3), dove viene visualizzata sullo schermo la scritta:

IL NUMERO A E' MAGGIORE DEL NUMERO B

Dopo di ciò il programma prosegue per il blocco 4), corrispondente all'istruzione di **STOP**, che blocca temporaneamente l'elaborazione.

Per proseguire l'elaborazione, ossia per confrontare fra loro altri due numeri, occorre eseguire il comando BASIC **CONT**.

Se invece nel blocco 2) il numero A risulta inferiore o eguale al numero B, ossia se è verificata la relazione A<=B, il programma viene posto di fronte ad una seconda alternativa, rappresentata dal blocco 5).

In questo blocco il programma si chiede se A<B (ossia se il numero A è inferiore al numero B).

In caso affermativo l'elaborazione prosegue per il blocco 6), dove viene

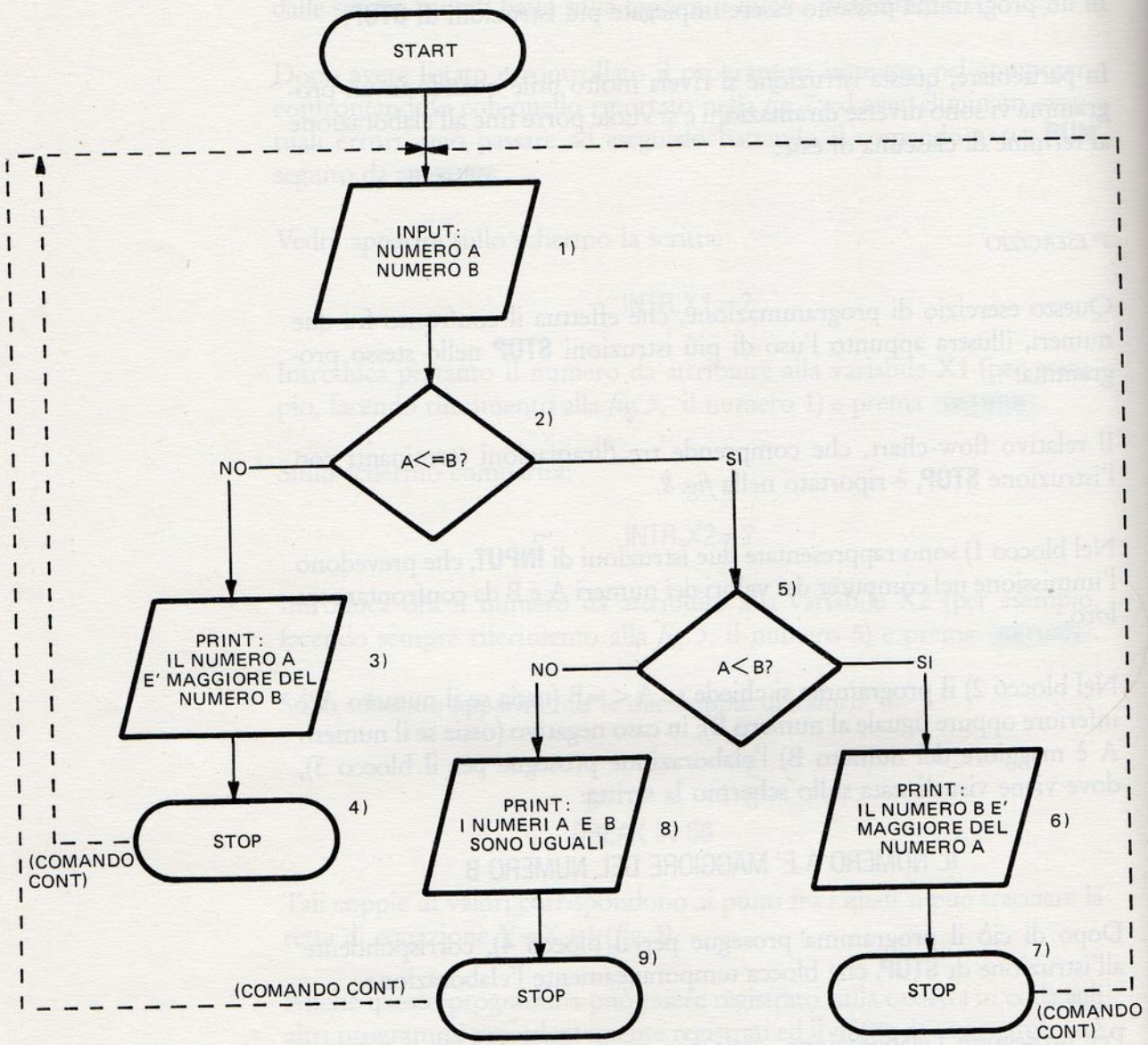


Fig. 8 – Flow-chart del programma che esegue il confronto tra due numeri.

visualizzato il messaggio:

IL NUMERO B E' MAGGIORE DEL NUMERO A

Dopo la visualizzazione del messaggio, l'elaborazione termina con il blocco 7), dove è presente la seconda istruzione di **STOP**.

Se invece nel blocco 5) non è verificata la relazione A < B, l'elaborazione

prosegue per il blocco 8), dove viene visualizzato il messaggio:

I NUMERI A E B SONO UGUALI

Dopo ciò viene eseguita l'istruzione di **STOP**, rappresentata dal blocco 9).

La codifica in linguaggio BASIC del flow-chart ora descritto è riportata nella *fig. 9*.

```
10 REM*******************************
20 REM* CONFRONTO FRA DUE NUMERI *
30 REM*******************************
40 INPUT"NUMERO A=";A
50 INPUT"NUMERO B=";B
60 IF A<=B THEN 90
70 PRINT"IL NUMERO A E'MAGGIORE DEL NUMERO B"
80 STOP:GOTO 40
90 IF A<B THEN 120
100 PRINT"I NUMERI A E B SONO UGUALI"
110 STOP:GOTO 40
120 PRINT"IL NUMERO B E'MAGGIORE DEL NUMERO A"
130 STOP:GOTO 40
140 END
```

Fig. 9 – Listato del programma che esegue il confronto tra due numeri.

Le istruzioni 4Ø e 5Ø prevedono l'introduzione dei numeri A e B da confrontare fra loro.

Nell'istruzione 6Ø il programma controlla se il numero A è inferiore o eguale al numero B: se tale condizione è verificata, salta all'istruzione 9Ø, dove è presente un'altra diramazione.

Se invece la relazione A<=B non è verificata, il programma prosegue con l'istruzione 7Ø, che provvede a visualizzare il relativo messaggio.

Dopo di ciò il programma si arresta con l'istruzione **STOP** alla linea 8Ø, per cui sullo schermo comparirà il messaggio:

BREAK IN 8Ø

L'elaborazione può essere ripresa, per confrontare fra loro altri due nu-

meri, scrivendo il comando BASIC:

CONT

Con questo comando il programma salta (**GOTO**) alla linea 4Ø, dalla quale riprende ad elaborare.

Nell'istruzione 9Ø vi è una seconda diramazione, nella quale il programma si chiede se A<B; se tale relazione è verificata, il programma prosegue per l'istruzione 12Ø e quindi si ferma con l'istruzione di **STOP** (13Ø) e sullo schermo comparirà il messaggio:

BREAK IN 13Ø

Se invece la relazione A<B non è verificata, il programma prosegue con l'istruzione 1ØØ e con l'istruzione 11Ø, dove si arresta con la visualizzazione sullo schermo del messaggio:

BREAK IN 11Ø

Esaminato il programma, può ora immetterlo nel computer battendo sulla tastiera le istruzioni che lo costituiscono.

Verifichi poi attentamente, come al solito, che non vi siano errori, quindi proceda alla sua esecuzione.

Per compiere questa operazione, può ad esempio effettuare il confronto dei due seguenti numeri: Ø.8, Ø.Ø3.

Esegua quindi il programma, scrivendo il comando **RUN** ed introducendo per A il valore Ø.8 e per B il valore Ø.Ø3.

Sullo schermo verranno visualizzate le seguenti scritte:

```
NUMERO A=  Ø.8
NUMERO B=  Ø.Ø3
IL NUMERO A E' MAGGIORE DEL NUMERO B
BREAK IN 8Ø
```

Per fare il confronto fra altri due numeri, batta il comando BASIC **CONT**.

Per esempio, introducendo per A il valore -55 e per B il valore 3, si otterrà sullo schermo la scritta:

```
NUMERO A= -55
NUMERO B=  3
IL NUMERO B E' MAGGIORE DEL NUMERO A
BREAK IN 130
```

Introducendo per A e per B lo stesso valore, ad esempio 7, compariranno sullo schermo le scritte che seguono:

```
NUMERO A=  7
NUMERO B=  7
I NUMERI A E B SONO UGUALI
BREAK IN 110
```

A conclusione dell'argomento, occorre fare notare che il comando **CONT** non può funzionare se sono state modificate o aggiunte istruzioni al programma, oppure se si è commesso un errore prima di fare ripartire il programma, o anche se il programma si è fermato a causa di un errore.

In questi casi il computer provvede a fare visualizzare sullo schermo il seguente messaggio d'errore:

```
CANT' CONTINUE
```
(non posso continuare)

* * * * * * *

In seguito...

Nella prossima lezione amplierà le Sue conoscenze di programmazione prendendo in considerazione le costanti, le variabili e le espressioni algebriche.

Eseguirà, inoltre, fra l'altro, un programma di calcolo algebrico ed un programma con l'istruzione IF nidificata.

CONTROLLO DIDATTICO 4

Domande

1) - Con quale istruzione BASIC si realizza un'alternativa semplice?

.....

.....

.....

.....

.....

.....

2) - Qual è il numero massimo di linee di programma utilizzato generalmente nei personal computer?

.....

.....

3) - Come può anche essere denominato, nei personal computer, il tasto **RETURN** ?

.....

.....

4) - Con il comando BASIC **CLEAR** il programma viene anche cancellato dalla memoria principale del computer?

.....

5) - Se, durante l'esecuzione di una istruzione **INPUT**, vengono forniti dati alfanumerici quando si aspettano dati numerici, quale messaggio d'errore appare sullo schermo?

.....

.....

.....

6) - Eseguire e visualizzare con un'istruzione ad esecuzione immediata (**PRINT**) il seguente calcolo:

$$\frac{2^5 + \left(\frac{4}{5} + 1{,}4^3\right)}{\frac{6}{7}\left(\frac{4}{5}+6\right)^{-2}}$$

.....

.....

7) - Scrivere ed eseguire con il computer il programma che prevede il calcolo del volume di un parallelepipedo retto. Come è noto, il volume del parallelepipedo è dato dalla formula: V=S1·S2·S3, dove S1, S2 e S3 sono le dimensioni dei tre spigoli. In questo programma devono quindi essere previste: tre distinte istruzioni **INPUT**, ognuna relativa alla dimensione di uno spigolo; un'istruzione che esegua il prodotto delle dimensioni dei tre spigoli ed assegni il risultato alla variabile V; infine due istruzioni di stampa (**PRINT**), una per visualiz-

zare la frase "IL VOLUME È" e l'altra per visualizzare il contenuto della variabile V stessa.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

8) - Modificando il programma previsto dalla domanda 7), creare un nuovo programma che consenta il calcolo del volume della piramide. Come è noto, il volume della piramide regolare si trova moltiplicando l'area della base (B) per l'altezza (h) e dividendo il prodotto per 3, ossia con la formula:

$$V = \frac{B \cdot h}{3} .$$

Verificare poi il risultato ottenuto eseguendo il programma sul computer.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

9) - Che cosa si intende con il termine gap?

.....

.....

.....

.....

10) - Come può essere scritta, in forma abbreviata, l'istruzione **PRINT**?

.....

.....

11) - Che cosa sarà visualizzato sullo schermo eseguendo la seguente istruzione?

PRINT "19";"84"

.....

12) - Senza consultare la lezione *Pratica del* BASIC *3*, disegnare qui sotto il flow-chart relativo al calcolo della somma di N numeri.

Risposte

1) - Una struttura alternativa semplice di programma si realizza in linguaggio BASIC mediante l'istruzione:

IF (1) **THEN** (2)

Nella suddetta istruzione al punto (1) si scrive una qualsiasi espressione di relazione (relazione che può essere algebrica o logica); al punto (2) si scrive il numero di riferimento, o linea di programma, dove salterà l'elaborazione nel caso che la relazione (1) risulti VERA.

2) - Il numero massimo di riferimento di un'istruzione è 65535; esso può essere inferiore, a seconda del tipo di personal computer.

3) - Il tasto **RETURN** viene anche denominato: **ENTER**, **CR**, **RET**, **RETN** e **CARRIAGE RETURN**.

4) - Battendo il comando BASIC **CLEAR**, il programma non viene cancellato dalla memoria del computer, ma soltanto dallo schermo.

5) - Se durante l'esecuzione di un'istruzione **INPUT** vengono forniti dati alfanumerici quando si aspettano dati numerici, sullo schermo appare il messaggio d'errore:

REDO FROM START
(rifare dall'inizio)

per cui occorre ribattere in modo corretto i dati.

6) - **PRINT** (2↑5+(4/5+1.4↑3))/(6/7*(4/5+6)↑-2)

Sullo schermo comparirà il risultato:

1917.48032

7) - Il programma per calcolare il volume del parallelepipedo è il seguente:

```
10 REM ****************
15 REM VOLUME DEL
20 REM PARALLELEPIPEDO
25 REM ****************
30 INPUT "SPIGOLO 1";S1
40 INPUT "SPIGOLO 2";S2
50 INPUT "SPIGOLO 3";S3
60 V = S1*S2*S3
70 PRINT "IL VOLUME E' :";
80 PRINT V
90 END
```

Questo programma può essere registrato sul lato B della cassetta ricevuta con la prima lezione (oppure su qualsiasi altra cassetta); il nome da assegnargli può essere "VOLUME 1".

8) - Programma per calcolare il volume della piramide:

```
10 REM ****************
15 REM VOLUME DELLA
20 REM PIRAMIDE
25 REM ****************
30 INPUT "BASE";B
40 INPUT "ALTEZZA";H
50 V = B*H/3
60 PRINT "IL VOLUME E' :";
70 PRINT V
80 END
```

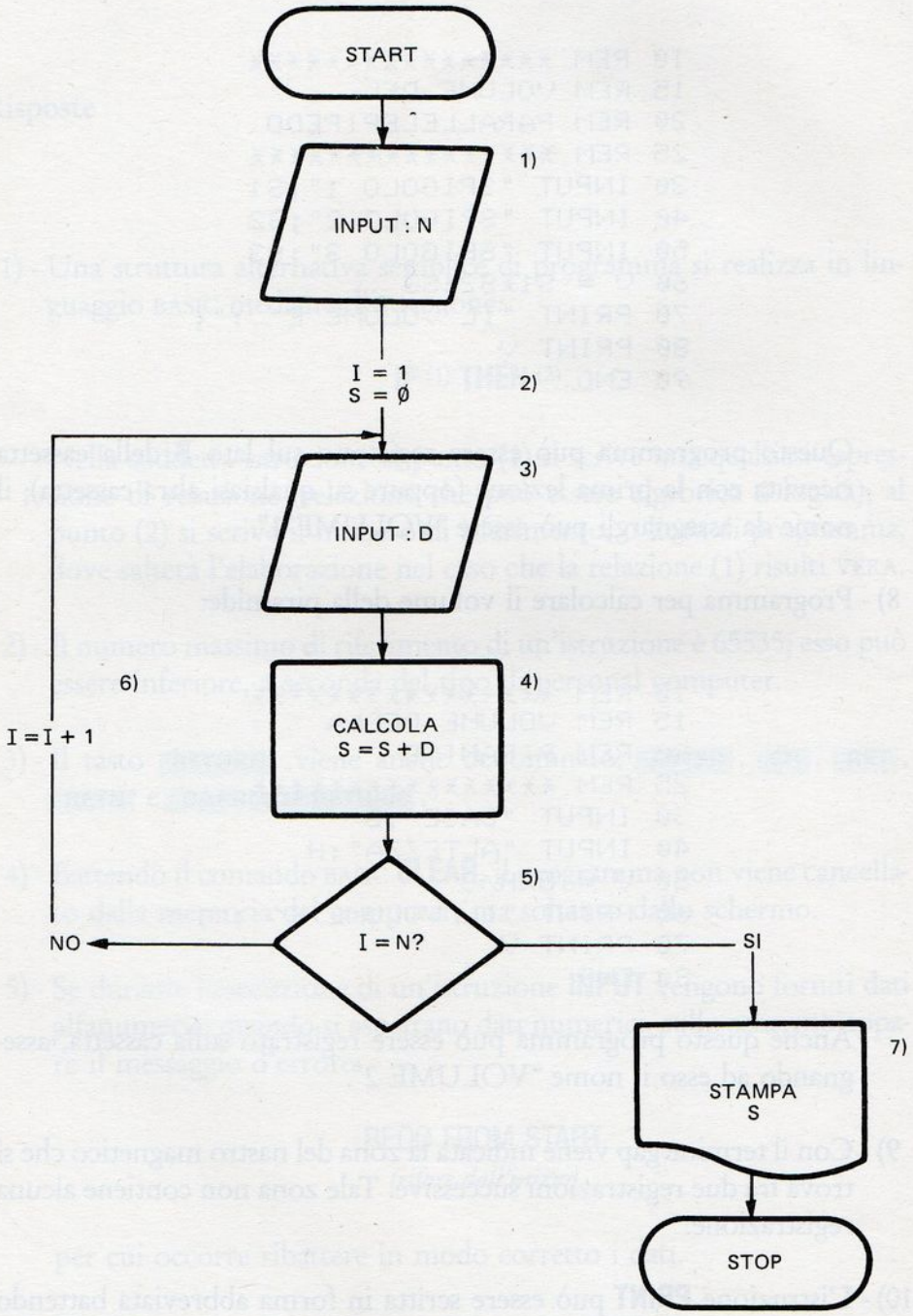
Anche questo programma può essere registrato sulla cassetta, assegnando ad esso il nome "VOLUME 2".

9) - Con il termine gap viene indicata la zona del nastro magnetico che si trova fra due registrazioni successive. Tale zona non contiene alcuna registrazione.

10) - L'istruzione **PRINT** può essere scritta in forma abbreviata battendo sulla tastiera il solo punto interrogativo (?).

11) - Eseguendo l'istruzione **PRINT** "19";"84", sullo schermo comparirà il numero 1984 perché il punto e virgola fa sí che i due numeri, 19 e 84, siano attaccati l'uno all'altro.

12) - Il flow-chart del calcolo della somma di N numeri è il seguente.



* * * * * * * *

PARLA BASIC

Corso teorico-pratico per corrispondenza di

ANALISI E PROGRAMMAZIONE BASIC

Scuola Radio Elettra

10126 Torino - Via Stellone 5

Tel. (011) 674432